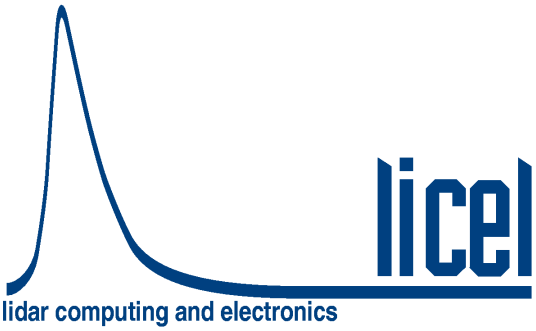


Licel Standard Resolution Multispectral Lidar Detector [SP32] – Installation and Reference Manual

Licel GmbH

April 16, 2015

Contents

1 The Licel Multispectral Lidar Detector **3**
- 1.1 Overview . . . 3
- 1.2 The Counter . . . 3
- 1.3 Detector . . . 4
  - 1.3.1 Gain characteristics . . . 4
  - 1.3.2 Crosstalk . . . 6
  - 1.3.3 Detector Homogenity . . . 7

2 Hardware Installation **9**
- 2.1 Power Supply Unit . . . 9
- 2.2 Detector & Electronics Housing . . . 10
- 2.3 Mounting the Detector & Electronics Housing . . . 10
- 2.4 Cable Connections . . . 10

3 Software Installation **12**
- 3.1 Preparation . . . 12
- 3.2 The Licel CD ROM . . . 13
- 3.3 Installing the Windows Applications . . . 14
- 3.4 Installing the Licel LabVIEW Libraries . . . 18

4 Setting up the Network **21**
- 4.1 Network Information . . . 21
- 4.2 Network Preparation . . . 22
  - 4.2.1 Establish the Connection . . . 22
  - 4.2.2 Diagnostics . . . 25
- 4.3 Network Setup . . . 26
  - 4.3.1 Fixed IP Address . . . 26
  - 4.3.2 DHCP Mode . . . 26
- 4.4 Reconfigure the PC . . . 27
- 4.5 TCP/IP Connection Parameters . . . 28
  - 4.5.1 TCP/IP Connection Problems (Software) . . . 30
- 4.6 Network Security . . . 31
  - 4.6.1 Changing the Administrator Password . . . 31
- 4.7 Hardware Reset . . . 32

5 Software Modules **33**
- 5.1 Overview . . . 33
- 5.2 PMT32 Pulse Height Distribution . . . 33
- 5.3 PMT32 Channel Acquisition . . . 36
- 5.4 PMT32 Viewer . . . 42
- 5.5 Calibration . . . 49

6 Data Acquisition using the Licel SP32 Detector **54**
6.1 Data Acquisition Overview . . . 54
6.1.1 SLAVE Mode . . . 54
6.1.2 PUSH Mode . . . 60
6.2 Transmitted Data Format . . . 65
6.2.1 SLAVE Mode . . . 65
6.2.2 PUSH Mode . . . 67
6.3 PUSH Mode Transmission Data Compression . . . 70
6.4 PUSH Data Time Stamps . . . 72
6.5 PUSH Mode Internal Shot Acquisition Status . . . 73
6.6 Efficient use of SP32 Detector for Data Acquisition . . . 73
6.7 Internal and External Data Averaging . . . 74
6.8 Client Side External Averaging of PUSH Data . . . 75

7 SP32 Controller TCPIP Command List and Syntax **78**

8 SP32 C Sources/Drivers **85**
8.1 SP32 Detector C Drivers . . . 85
8.1.1 licel_sp . . . 85
8.1.2 licel_sp_util . . . 87
8.1.3 licel_sp_tcpip . . . 88
8.2 SP32 Detector Sample Applications . . . 90
8.2.1 SP32_SlaveMode_Acquisition_Application . . . 90
8.2.2 Application Configuration File . . . 91
8.3 Data File format . . . 93
8.3.1 Sample file header . . . 93

9 LabVIEW Driver VIs **95**
9.1 Licel TCPIP Driver VIs . . . 95
9.1.1 Top Level VIs . . . 95
9.1.2 Controller related VI's . . . 96
9.1.3 Transient Recorder Commands applicable to PM32 Detector . . . 97
9.1.4 PMT Commands applicable to PM32 Detector . . . 97
9.1.5 Network Security . . . 98
9.2 PM32 Detector Specific VIs . . . 99

Chapter 1

The Licel Multispectral Lidar Detector

1.1 Overview

The Licel Multispectral Lidar Detector allows simultaneous detection of multiple spectrometer wavelengths. It is based on a multianode, metal-channel-dynode photomultiplier. 32 photocathode elements together with 32 single photon counting systems provide 2-dimensional, spectral and range resolved data.

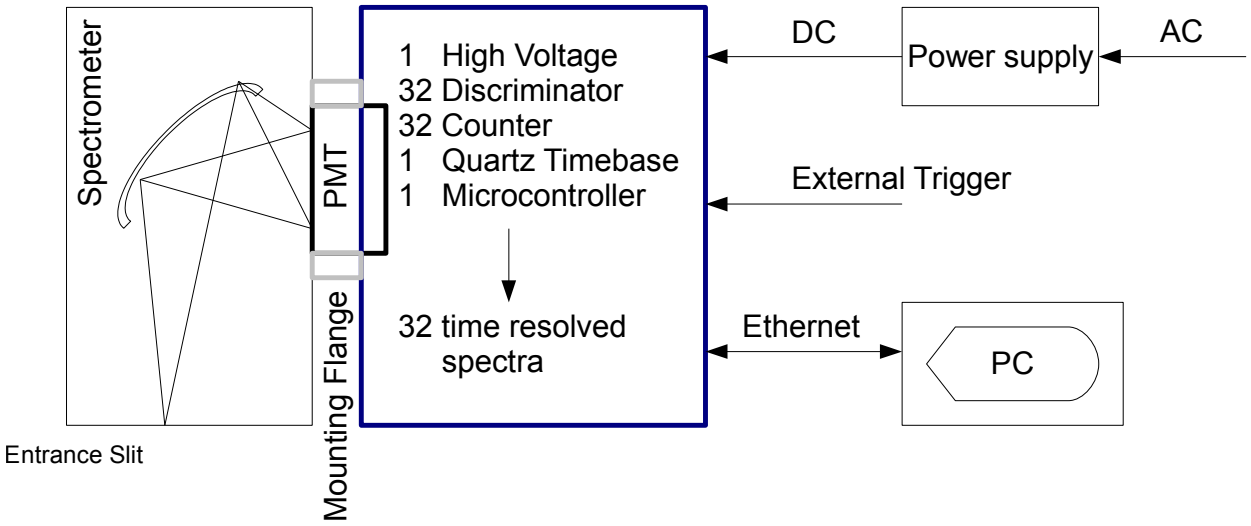


Figure 1.1: Licel Multispectral Lidar Detector

The high voltage supply, an adjustable discriminator and the trigger logic is integrated on the module. The Ethernet interface is used together with LabVIEW software to control the measurement and readout the acquired data. The Licel Multispectral Lidar Detector can be used with various flat field spectrographs.

1.2 The Counter

The counter contains a microcontroller which receives its commands over the Ethernet. It will set the high voltage for the PMT and the discriminator level. It will transmit the averaged time resolved spectra over the Ethernet to the PC.
The 32 channel counter receives the raw anode signals from the PMT. It also provides the software controlled high voltage for the PMT. The 32 raw signals are then amplified and fed in to the discriminator. The threshold level is software controlled. Behind the 32 discriminators are 32 counters which

will count the single photons for each channel individually.
Beginning with an external trigger the counts of all 32 channels are recorded. A fixed quartz based time base will be used to read out all 32 counters in parallel. The microcontroller will average the spectra over multiple trigger events. Once enough data has been acquired the summed up spectra will be transferred to the PC.

1.3 Detector

The module is based on a Hamamatsu H7260 Multi Anode photomultplier. 32 photocathodes, each 0.8 mm $\times$ 1 mm have their own dynode structure to amplify photoelectrons from each cathode strip. The mechanical outline is shown in fig. 1.2.

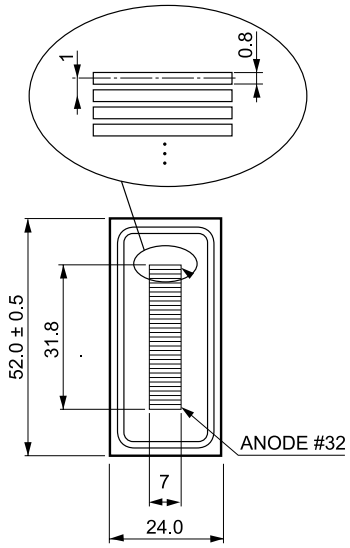


Figure 1.2: Mechanical outline of the detector cathodes

1.3.1 Gain characteristics

Like in individual photomultipliers, these dynode chains vary with respect to their individual gain-voltage behavior. The gain of each cathode strip depends on the voltage setting and on these gain characteristics. The pulse height distributions for all dynode chains at 900 V and 950 V are shown in the plots below.

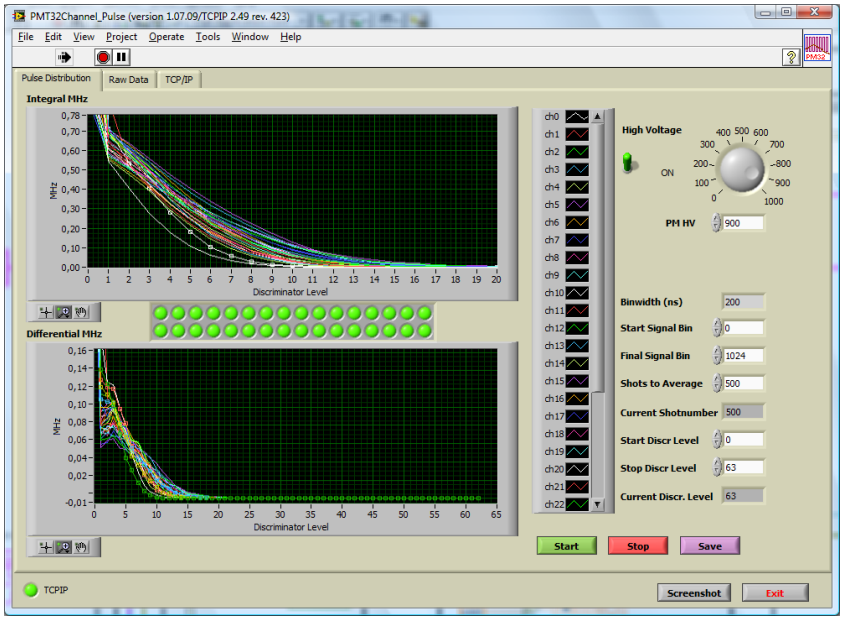


Figure 1.3: Pulse height distribution at 900 V

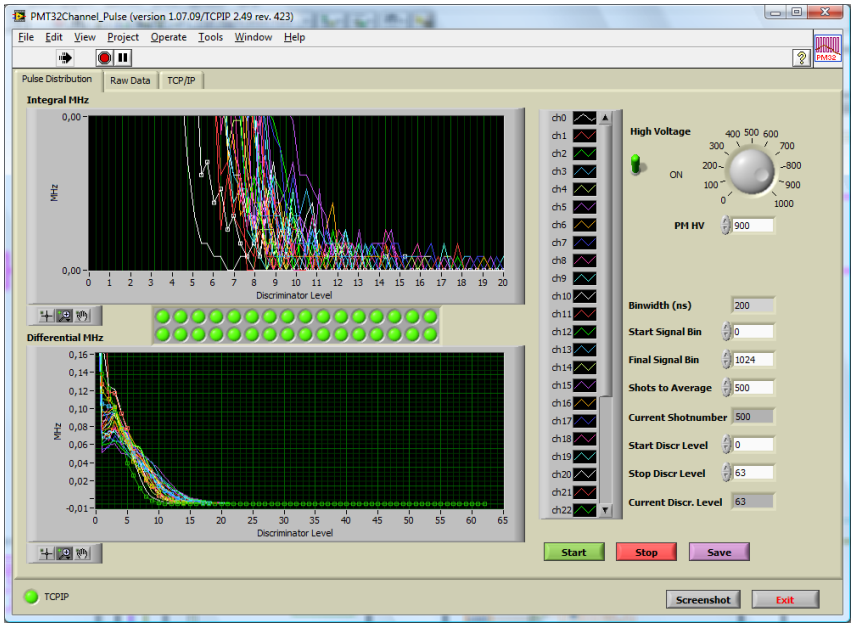


Figure 1.4: Pulse height distribution (zoomed) at 900 V

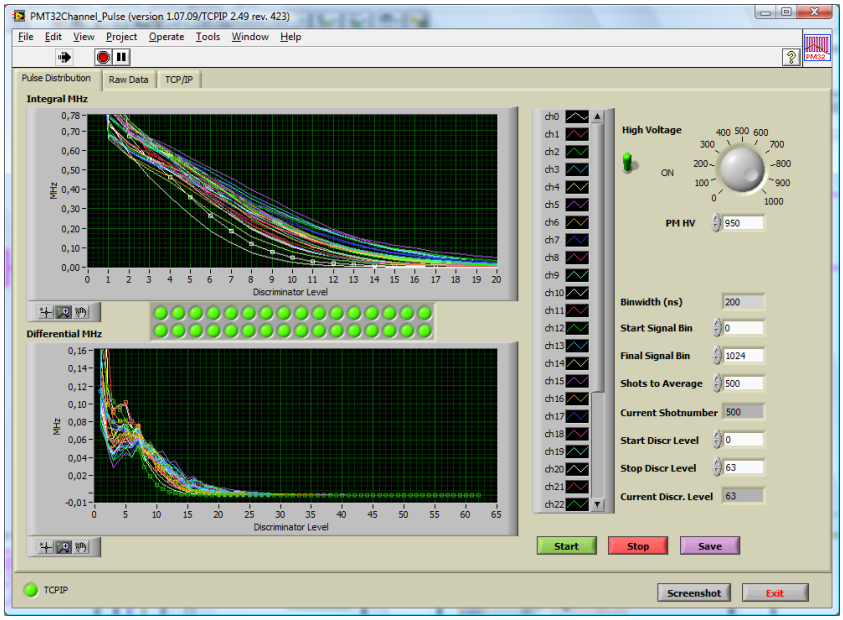


Figure 1.5: Pulse height distribution at 950 V

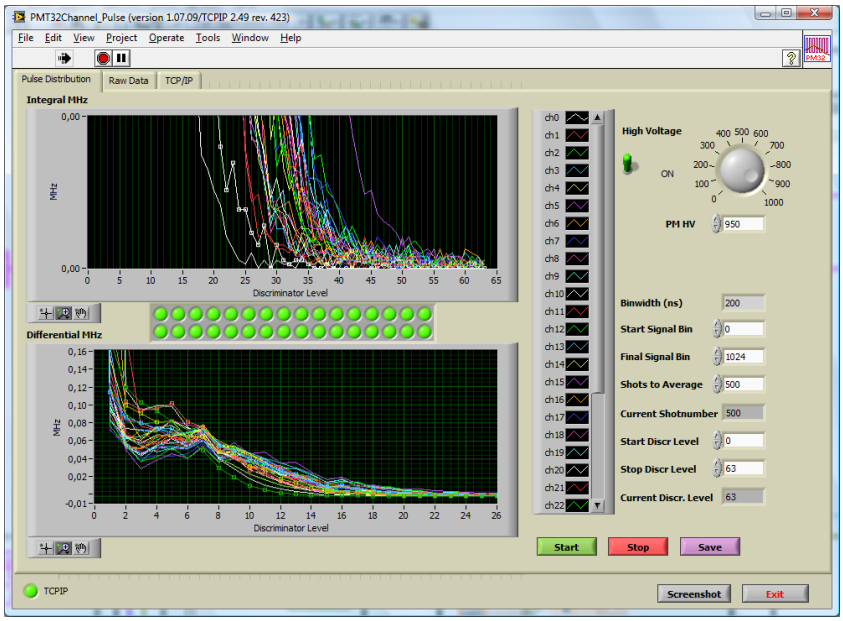


Figure 1.6: Pulse height distribution (zoomed) at 950 V

1.3.2 Crosstalk

Two different effects are responsible for crosstalk effects in a multispectral application using the Licel SP-32 detector assembly at the spectrometer image plain.

1. Straylight from the grating. In a single grating spectrometer typical straylight levels are in the order of 10^{-4}.

2. Amplification of photoelectrons from neighboring photocathodes. If photoelectrons are emitted towards the neighbouring dynode chain, these photoelectrons can be amplified by the ”wrong” dynode chain. For the H7260 detector this probability is 3% for the adjacent channel and 0.6% for the n+2nd channel.

3. Saturation effects will occur when too many photoelectrons or amplified secondary electrons

are created to be amplified by the individual dynode chain. As soon as one dynode chain is saturated, the electric field towards the next dynode will decrease and the neighboring dynodes will amplify these electrons. In fig. 1.7 a square wave pulse, saturating dynode chain #16 and #17 appears on all dynodes. Increasing the light level even further leads to crosstalk effects on all dynodes at the beginning of the square wave pulse, followed by a breakdown of the amplification chain. This is demonstrated in fig. 1.8.

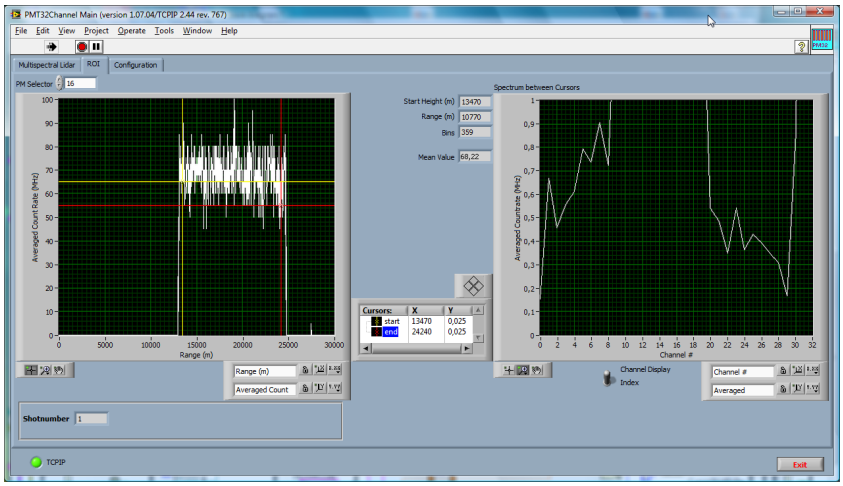


Figure 1.7: Display for a square wave pulse, saturating dynode chain #16 and #17 appearing on all dynodes

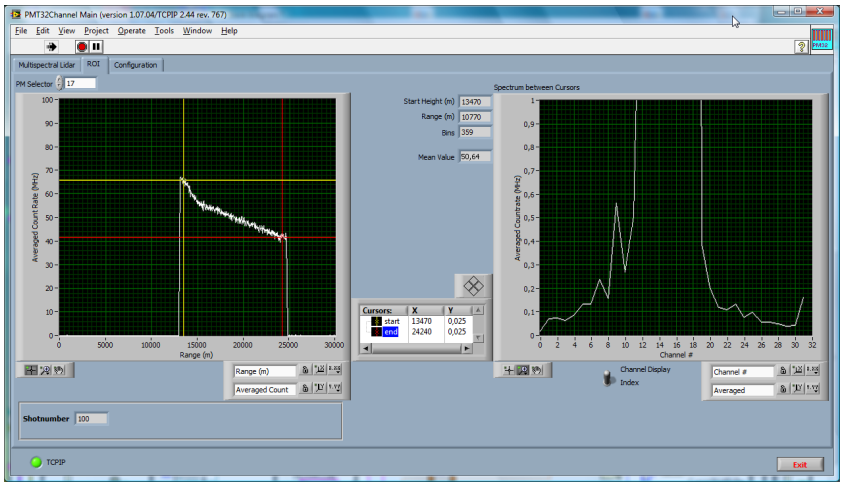


Figure 1.8: Example for crosstalk effects on all dynodes at the beginning of the square wave pulse, followed by a breakdown of the amplification chain

The max. average current per anode is 6µA. This corresponds to 12.5×10^6 photoelectrons per second at +850 V (3×10^6 gain). Like in other photomultipliers the max. average current value can be exceeded for a short period of time, but exceeding this value by more than a factor of 10...20 is not recommended to avoid saturation effects as described above. This means that light levels corresponding to single photon count rates of 125MHz to 250MHz should be avoided.

1.3.3 Detector Homogenity

Typical sensitivity variations for the Hamamatsu H7260 detectors are shown in the H7260 series datasheet attached here. The individual gain variations for the detector ZB1487 are shown in fig. 1.9.

This sensitivity curve has been determined by illuminating the detector through a Newport MS125 spectrometer and varying the incident angle of the grating. The measurement was taken at HV 900 V and a discriminator setting of 4 and 8.

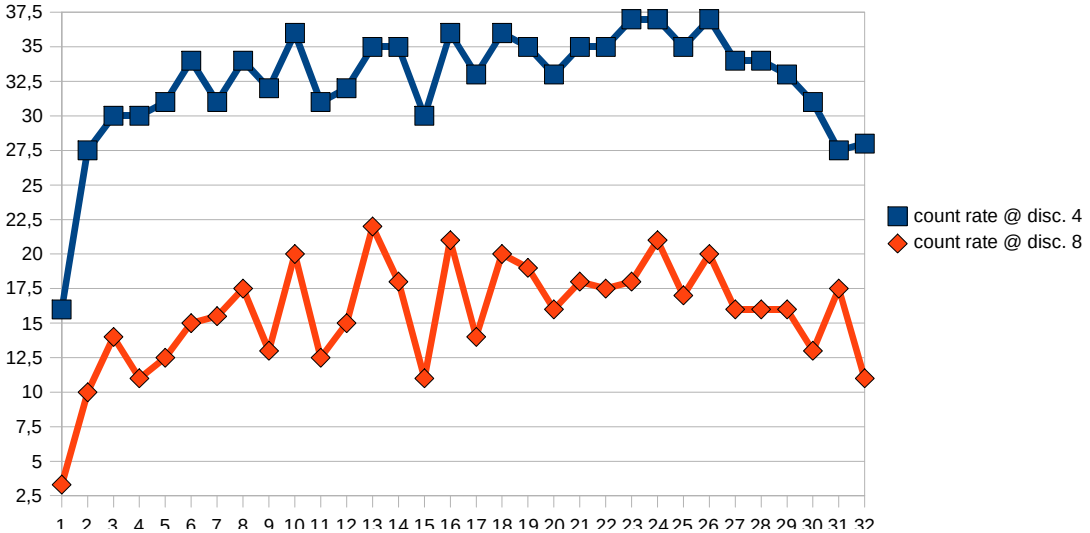


Figure 1.9: Individual sensitivity of the 32 cathode portions including their dynode chain

Chapter 2

Hardware Installation

The Licel Multispectral Lidar Detector consists of the following equipment:

1. power supply unit
2. detector & electronics housing
3. mounting flange (premounted at the detector & electronics housing)
4. DC power cable (Lemo 4 pole)
5. line cable
6. Ethernet cross link cable
7. trigger cable (Lemo CAMAC to BNC coaxial)

2.1 Power Supply Unit

The power supply cassette can be integrated to standard 19” form factor systems. The line connector is located at the rear side. The power switch, the control LED, and the DC output connector are on the front side.

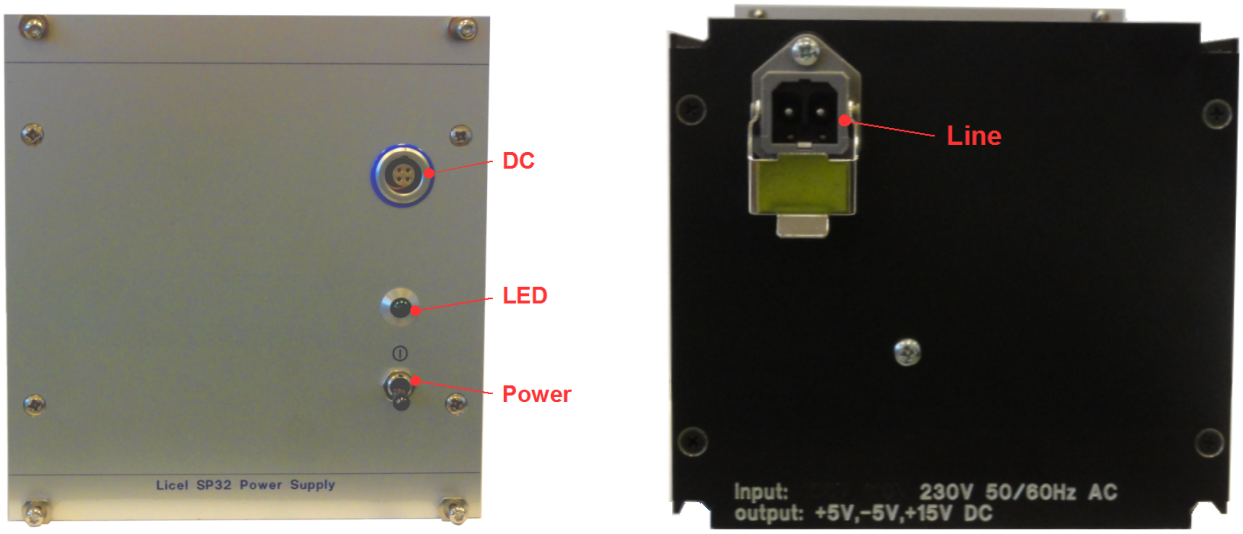


Figure 2.1: Front (left) and rear sides of the power supply cassette

2.2 Detector & Electronics Housing

The detector & electronics housing contains the 32 channel detector, the 32 channel counter and the network interface. On top the DC input connector (4 pole Lemo), the network reset switch, the trigger input (Lemo CAMAC), and the Ethernet connector (RJ45) are located as seen in the figure.

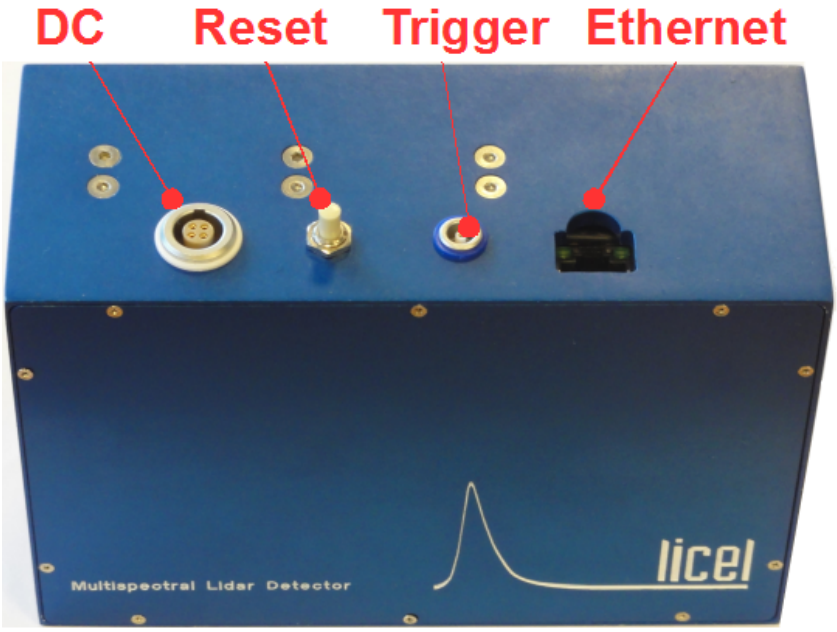


Figure 2.2: Detector & electronics housing

2.3 Mounting the Detector & Electronics Housing

To connect the detector & electronics housing to the spectrometer the mounting flange is used. Mounting flanges for several spectrometers are available. The appropriate mounting flange must be specified when ordering the Licel Multispectral Lidar Detector.
The mounting flange

- is already premounted at the detector & electronics housing
- makes sure that the photocathode plane of the detector lies in the focal plane of the spectrometer
- uses O-ring seals at it's mounting surfaces
- has prepared holes and/or threads for mounting at the spectrometer.

Please make sure that daylight illumination of the detector is carefully avoided during mounting. The holes and/or threads of the mounting flange have to be aligned to the spectrometer's threads and/or holes. Then it must be carefully screwed.

2.4 Cable Connections

The following cable connections are required:

1. line connection to the rear side of the power supply unit (line cable included)

2. DC supply connection from the front of the power supply to the detector & electronics housing (Lemo 4 pole cable included)
3. trigger input from a trigger source to the trigger input at the detector & electronics housing (Lemo CAMAC to BNC cable included)
4. Ethernet connection to the detector & electronics housing. A cross link cable for a direct connection to the computer is provided. A straight patch cable is required when connecting to an Ethernet switch or hub.

Chapter 3

Software Installation

Licel provides a package of software modules for setting up the Licel Multispectral Lidar Detector for network operation, and for operating the Licel Multispectral Lidar Detector. These software modules are written in LabVIEW's G language. The software is provided as LabVIEW source for users who have LabVIEW (beginning with version 8.2) installed, or alternatively as a set of Windows applications. The Windows applications come within a Windows Installer package for an easy installation on your Windows (2000|XP|Vista) computer. Licel provides the software on a CD ROM.

3.1 Preparation

Windows Application Users

If you have used older versions of Licel Windows applications it is recommended to backup existing initialization files (`*.ini`).

Search the installation directory of the older version of Licel Windows applications (standard: `<Program Files Directory>\Licel`) and backup all files with the ending `*.ini` to an archive file (zip, ARJ, TAR, etc...) or onto a CD ROM.

LabVIEW Users

If you have used older versions of Licel LabVIEW libraries it is necessary to remove and backup older versions.

1. Backup all your current Licel software libraries, in case you want to restore them, by either compressing them (zip, ARJ, TAR, etc...) or burning them onto a CD ROM.
2. Scan your disks to find all versions of the following files and delete them once you have made backups of them

```
Advanced Viewer.llb
Datafile.llb
Licel Graph.llb
Licel Module.llb
Licel TCPIP.llb
Licel Util.llb
PMT32 Viewer.llb
PMT32Channel Pulse.llb
PMT32Channel.llb
TCPIP-Acquis.llb
```

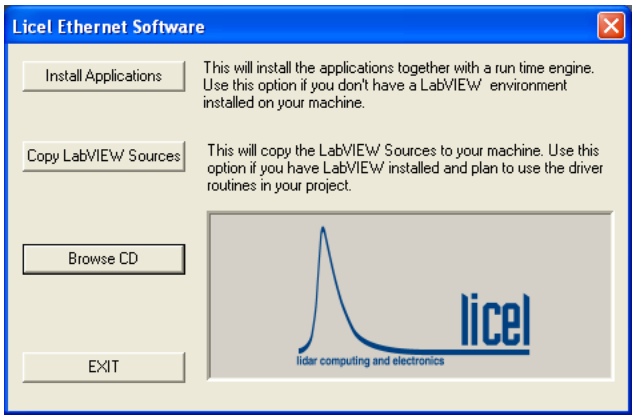
Please note: Licel may have provided individual software solutions with additional or less LabVIEW library files than noted in the list above.

3. Search the directory your older version of Licel LabVIEW libraries reside and backup all initialization files (`*.ini`).

3.2 The Licel CD ROM

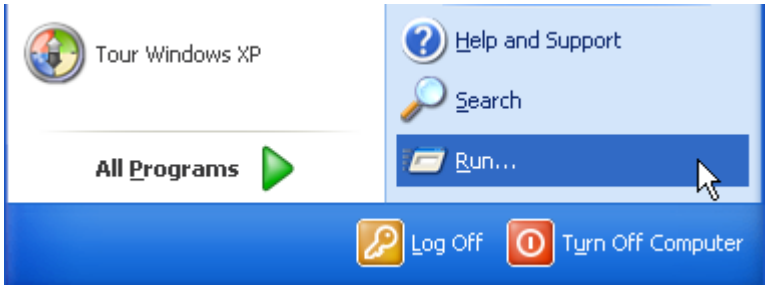
The CD ROM provided by Licel contains both, the LabVIEW sources and the Windows Installer for installing the Windows applications, and a documentation folder. Licel may add customer specific components on the CD ROM.

1. Insert the Licel CD into your CD ROM drive.
2. The following selection dialog should appear automatically:

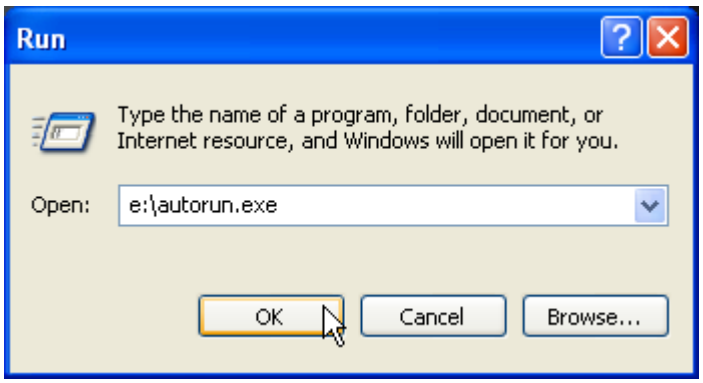

Licel Ethernet Software
Install Applications
This will install the applications together with a run time engine. Use this option if you don't have a LabVIEW environment installed on your machine.
Copy LabVIEW Sources
This will copy the LabVIEW Sources to your machine. Use this option if you have LabVIEW installed and plan to use the driver routines in your project.
Browse CD
EXIT
licel
lidar computing and electronics

   - Press *Install Applications* to start the Windows Installer which will guide you through the installation of the Licel Applications. Please proceed to the section 3.3.
   - Press *Copy LabVIEW Sources* to copy the LabVIEW source files from the CD ROM to a folder of your choice on your computer. Please note the remarks according to existing LabVIEW library files. Please refer to the section 3.4 for further details.
   - Press *Browse CD* to inspect the content of the CD ROM
   - Press *Exit* to exit the dialog without any further action.

3. If the selection dialog does not come up, please press the start button, select **Run** in the upcoming menu:



   In the dialog box enter `<CD drive letter>:\autorun.exe` where `CD drive letter` is the letter corresponding to your CD ROM drive (`E` in the picture), and press *OK*.



The selection dialog from above should now be started, you will be able to select one of the installation options.

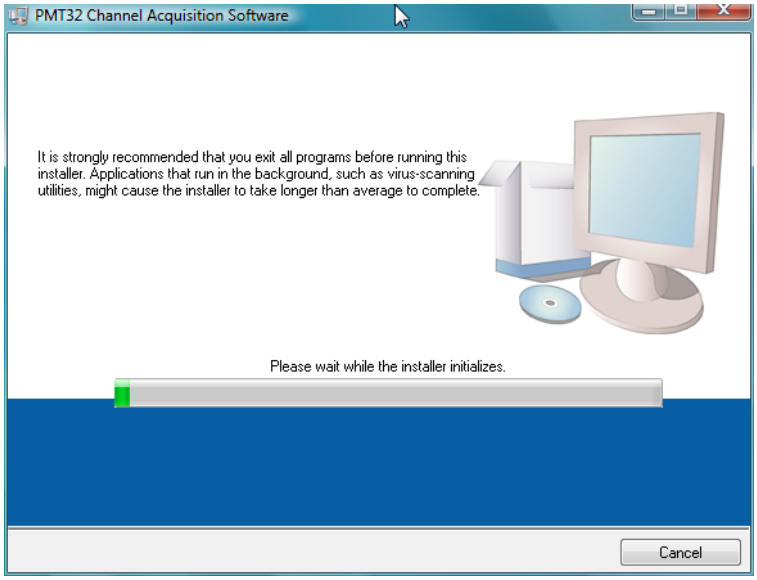
3.3 Installing the Windows Applications

This subsection describes the installation process for the Licel Windows applications. To operate the Licel Windows applications a LabVIEW runtime environment needs to be installed. The Windows applications together with the LabVIEW runtime environment come as a Windows Installer package. For the installation of the LabVIEW runtime part of the installer package local administrator privileges are required.

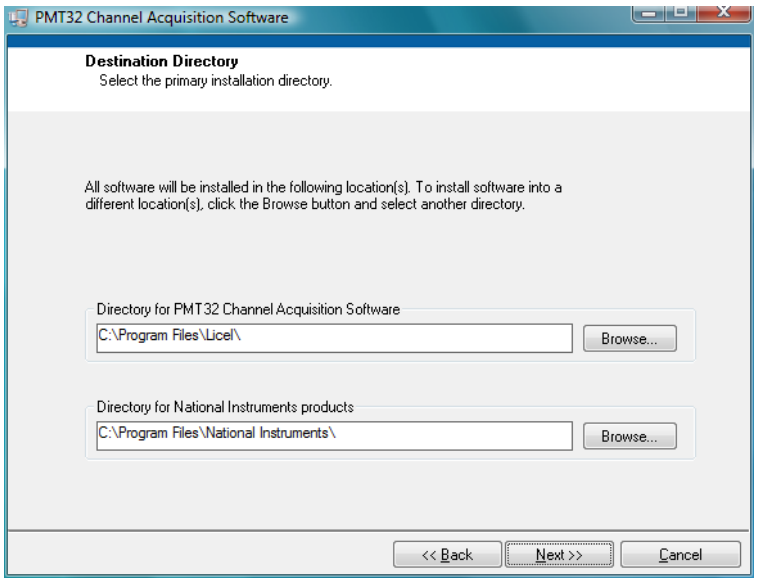
The following items describe the installation process after starting the Windows Installer's setup routine (`setup.exe`). The setup program is automatically started when using the CD ROM and pressing **Install Applications** in the setup selection dialog. `setup.exe` is located on the Licel CD ROM in the subdirectory `Installer` or in the temporary directory you unzipped the downloaded Licel Installer package. You may directly start the setup routine from the corresponding directories.

The Windows Installer dialogs will guide you through the installation process.

1. The installation will start with the following screen. Please wait until the installer has initialized.

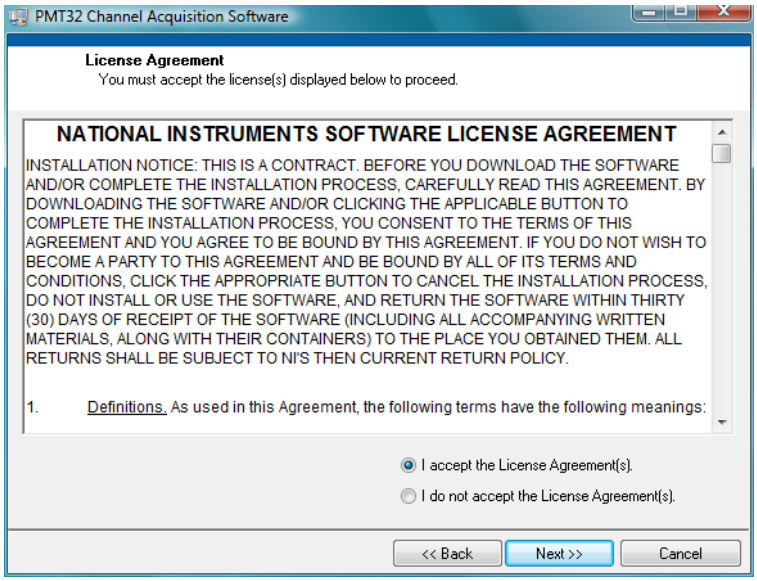

PMT32 Channel Acquisition Software
It is strongly recommended that you exit all programs before running this installer. Applications that run in the background, such as virus-scanning utilities, might cause the installer to take longer than average to complete.
Please wait while the installer initializes.
Cancel

2. The next screen will allow you to choose the installation directory (standard: `<Program Files Directory>\Licel`) and specify the directory where former National Instruments products are installed. The latter directory will be the root directory for the LabVIEW runtime engine.

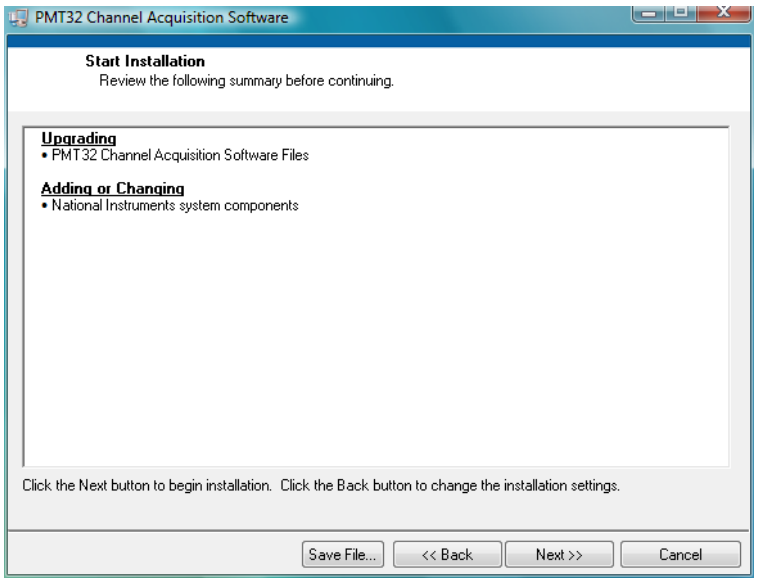

PMT32 Channel Acquisition Software
Destination Directory
Select the primary installation directory.
All software will be installed in the following location(s). To install software into a different location(s), click the Browse button and select another directory.
Directory for PMT32 Channel Acquisition Software
C:\Program Files\Licel\
Browse...
Directory for National Instruments products
C:\Program Files\National Instruments\
Browse...
<< Back
Next >>
Cancel

If you would like to change one of these directories click **Browse** and choose or create a directory of your choice. Click **Next** to proceed.

3. As National Instrument's LabVIEW runtime engine has to be installed you are asked to accept a software license agreement for National Instruments software. Plese check *I accept the license agreenment(s)* and proceed by clicking **Next**

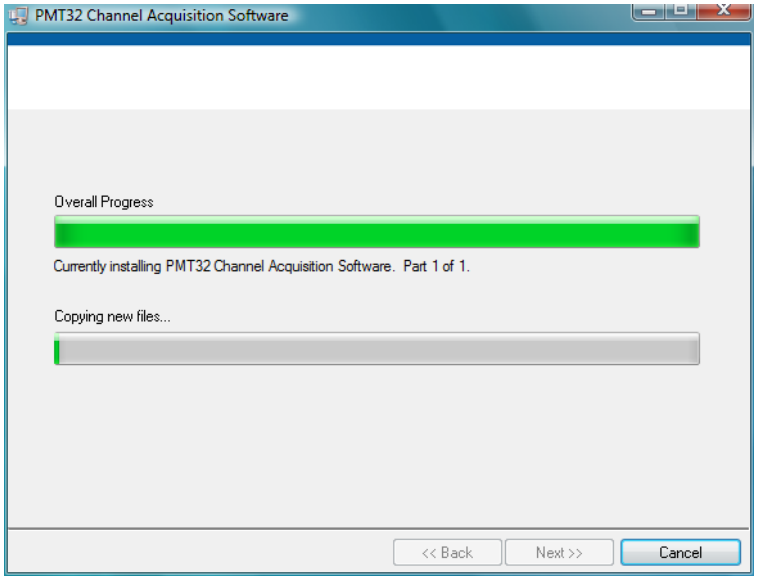

PMT32 Channel Acquisition Software
License Agreement
You must accept the license(s) displayed below to proceed.
NATIONAL INSTRUMENTS SOFTWARE LICENSE AGREEMENT
INSTALLATION NOTICE: THIS IS A CONTRACT. BEFORE YOU DOWNLOAD THE SOFTWARE AND/OR COMPLETE THE INSTALLATION PROCESS, CAREFULLY READ THIS AGREEMENT. BY DOWNLOADING THE SOFTWARE AND/OR CLICKING THE APPLICABLE BUTTON TO COMPLETE THE INSTALLATION PROCESS, YOU CONSENT TO THE TERMS OF THIS AGREEMENT AND YOU AGREE TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT WISH TO BECOME A PARTY TO THIS AGREEMENT AND BE BOUND BY ALL OF ITS TERMS AND CONDITIONS, CLICK THE APPROPRIATE BUTTON TO CANCEL THE INSTALLATION PROCESS, DO NOT INSTALL OR USE THE SOFTWARE, AND RETURN THE SOFTWARE WITHIN THIRTY (30) DAYS OF RECEIPT OF THE SOFTWARE (INCLUDING ALL ACCOMPANYING WRITTEN MATERIALS, ALONG WITH THEIR CONTAINERS) TO THE PLACE YOU OBTAINED THEM. ALL RETURNS SHALL BE SUBJECT TO NI'S THEN CURRENT RETURN POLICY.
1. Definitions. As used in this Agreement, the following terms have the following meanings:
I accept the License Agreement(s).
I do not accept the License Agreement(s).
<< Back
Next >>
Cancel

4. The next screen will inform about the software components that will need to be installed on your system.

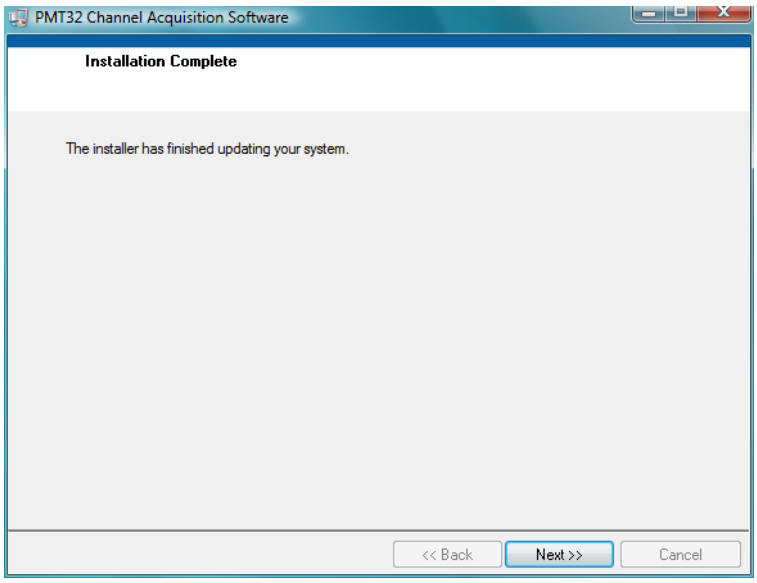

PMT32 Channel Acquisition Software
Start Installation
Review the following summary before continuing.
Upgrading
• PMT32 Channel Acquisition Software Files
Adding or Changing
• National Instruments system components
Click the Next button to begin installation. Click the Back button to change the installation settings.
Save File...
<< Back
Next >>
Cancel

Please click **Next** to proceed.

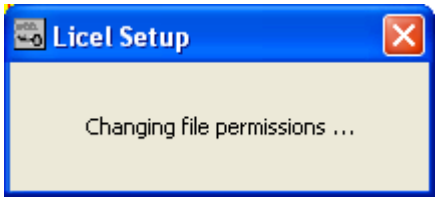
5. The next screen contains the progress indicators during the installation.



6. When the installation has finished please click **Next** to proceed.

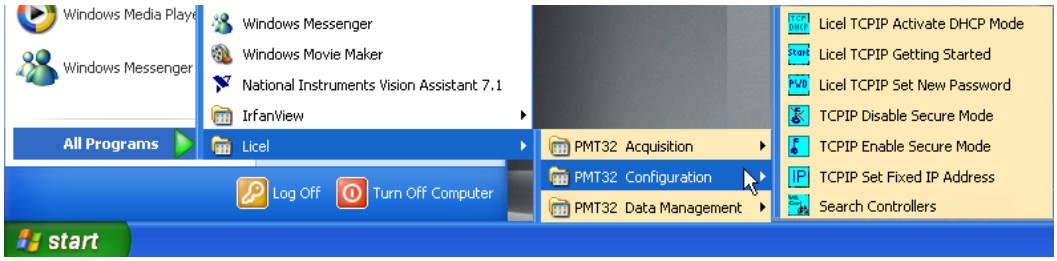


7. At the end of the installation process permissions on some files will be set.

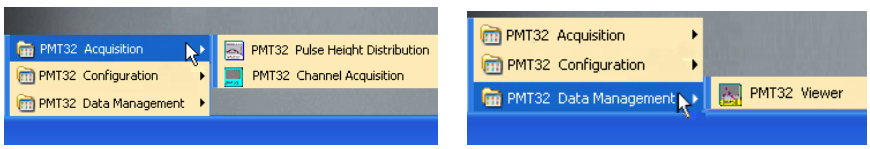


8. If it is is necessary to restart the computer after the applications a dialog will pop up with an appropriate button for the restart procedure.

9. After the installation has successfully been completed you are able to start the Windows applications through the corresponding entry in the program group **Licel** in the Windows start menu:

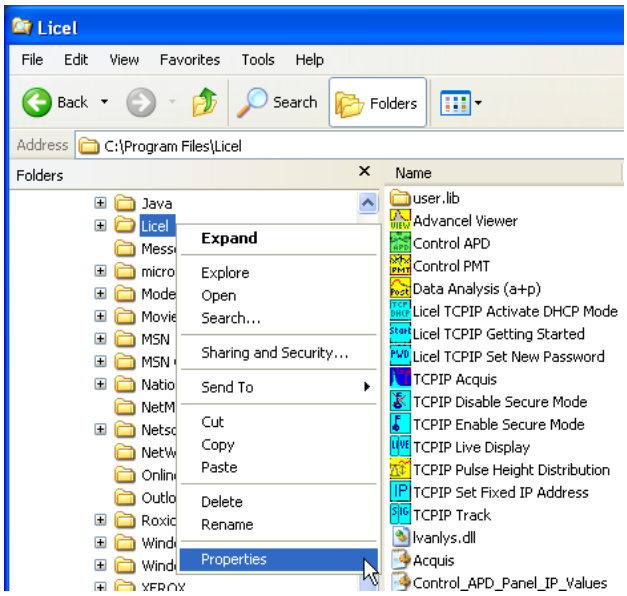


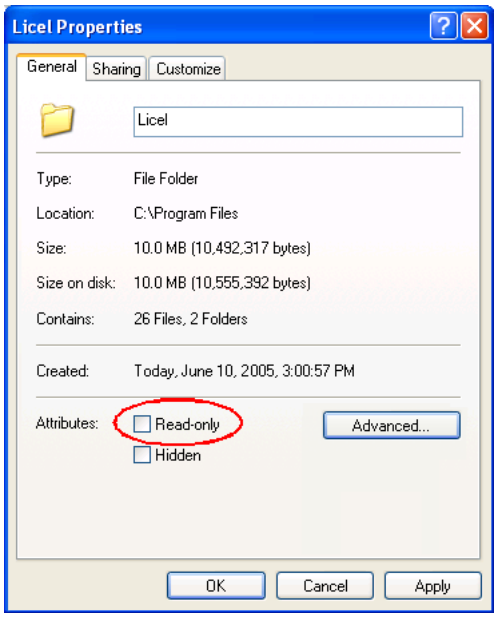
The links to the applications are grouped into the subfolders **PMT32 Configuration**, **PMT32 Acquisition**, and **PMT32 Data Management**.



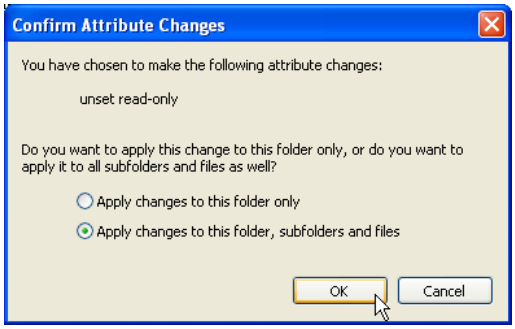
10. Please note that the Licel software needs write permissions for the initialization files located in the installation directory. Normally sufficient permissions are set during the installation process.

If any problems indicating missing permissions occur select the directory and right-click on it. Select **Properties** from the context menu.





Verify that the `"Read-only"` attribute is not checked, uncheck it if necessary. Click *OK* and check in the next dialog **Apply changes to this folder, subfolders and files**. Leave the dialog by clicking *OK*


Confirm Attribute Changes
You have chosen to make the following attribute changes:
unset read-only
Do you want to apply this change to this folder only, or do you want to apply it to all subfolders and files as well?
Apply changes to this folder only
Apply changes to this folder, subfolders and files
OK
Cancel

11. If you have backuped your initialization files from an older version of Licel Ethernet Software you may copy the TCP/IP parameters from the corresponding old initialization files to the files of the current installation. Please note that copying information from older to new initialization files should be done value by value (line by line).

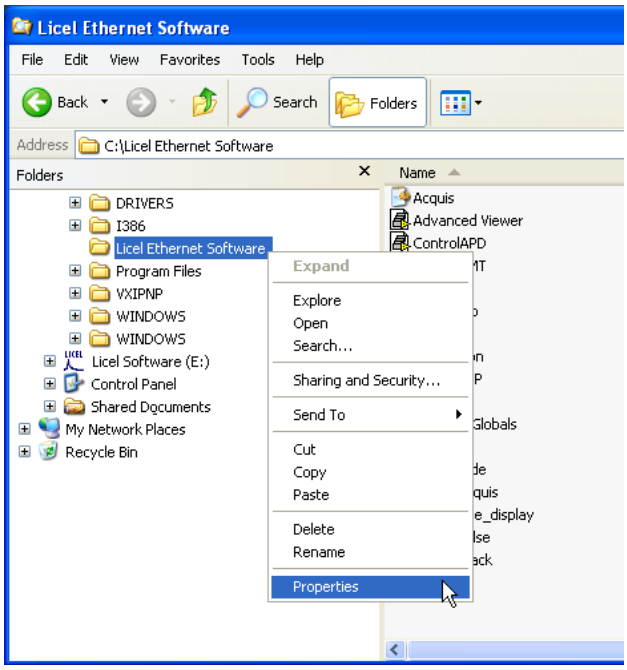
3.4 Installing the Licel LabVIEW Libraries

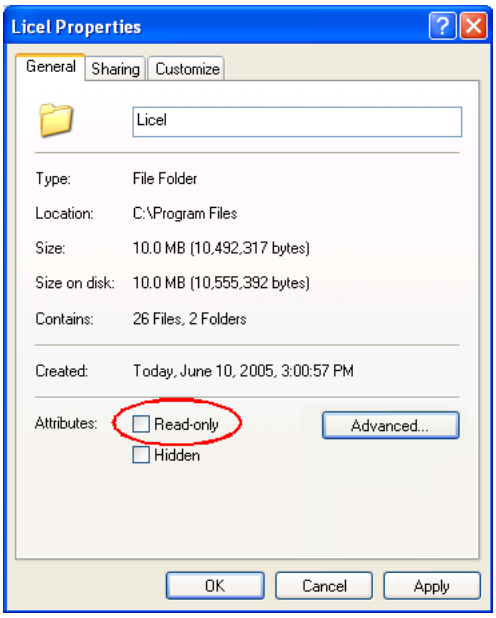
To install the Licel LabVIEW libraries you may choose between the following options:

- The Licel LabVIEW Libraries will be copied automatically from the Licel CD ROM by pressing **Copy LabVIEW Source** in the setup selection dialog. You will be asked to select or create a target folder.
- You may manually copy all files contained in the directory `LabVIEW Files` on the CD ROM to a directory of your choice.

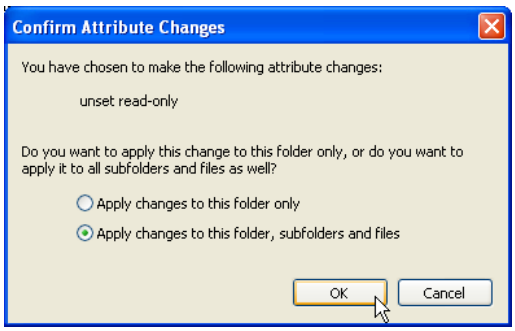
- If you downloaded the Licel software from http://www.licel.com/software.htm please unpack the content from the downloaded zip file and copy it to a directory of your choice.

Please note that in the case the software is copied from a CD you may have to unselect the *"Read-only"* attribute for the destination folder. This is done by selecting the directory and right-clicking on it. Select **Properties** from the context menu.



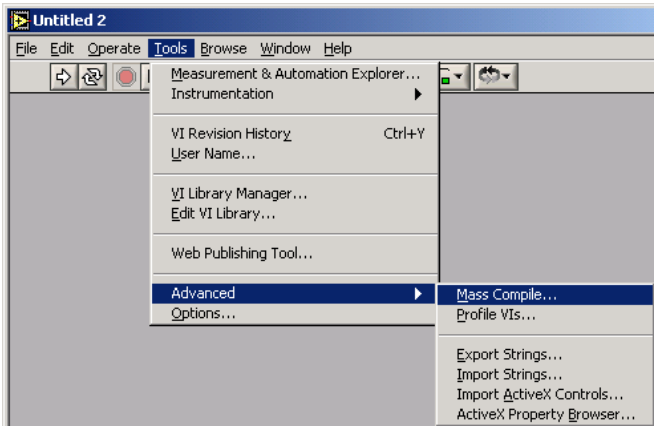


Verify that the "`Read-only`" attribute is not checked, uncheck it if necessary. Click *OK* and check in the next dialog *Apply changes to this folder, subfolders and files*. Leave the dialog by clicking *OK*

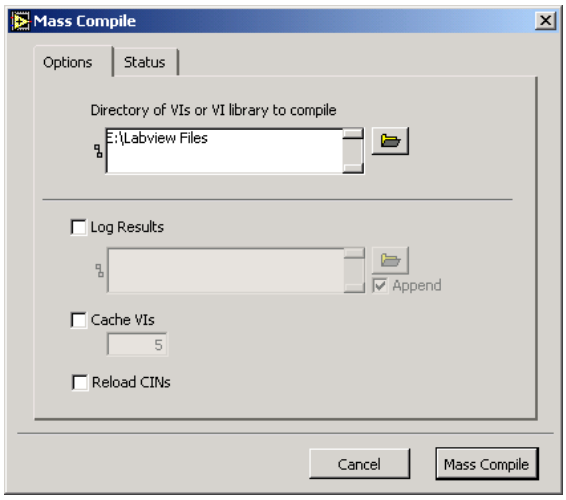

Confirm Attribute Changes
You have chosen to make the following attribute changes:
unset read-only
Do you want to apply this change to this folder only, or do you want to apply it to all subfolders and files as well?
Apply changes to this folder only
Apply changes to this folder, subfolders and files
OK
Cancel

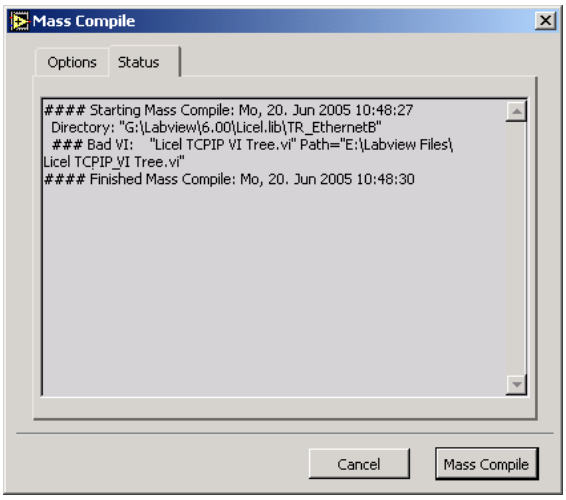
Now you should be able to run all the files. If you are still having problems, apply a mass compile to the directory where the software was extracted to:

1. Start LabVIEW, press on *New VI*.
2. Select the menu **Tools**, then **Advanced**, and finally **Mass Compile...**.



3. You will be asked to select a directory, select the target directory of the LabVIEW source files.
4. Press *Mass Compile* in the next dialog.




Mass Compile
Options
Status
Starting Mass Compile: Mo, 20. Jun 2005 10:48:27
Directory: "G:\Labview\6.00\Licel.lib\TR_EthernetB"
Bad VI: "Licel TCPIP VI Tree.vi" Path="E:\Labview Files\Licel TCPIP VI Tree.vi"
Finished Mass Compile: Mo, 20. Jun 2005 10:48:30
Cancel
Mass Compile

5. Later the mass compile status will be shown. Please ignore that the vi `Licel TCPIP VI Tree` is indicated to be a ”bad vi”.

Please note that the removal of older libraries is a necessity, since LabVIEW often links to various libraries with the same name. As a result, if a library is installed twice, one can not be certain which library is actually being used.
If you still have any problems, please contact Licel for further assistance.

Chapter 4

Setting up the Network

To operate the Licel Multispectral Lidar Detector in your local network you will have to carry out the following steps as described in the corresponding subsections:

1. Get the required **Network Information**.
2. Prepare the PC to communicate with the ethernet controller using a cross-link cable (**Network Preparation**).
3. Setup the ethernet controller for your local area network either by setting a fixed IP address or by activating the DHCP mode (**Network Setup**).
4. **Reconfigure the PC** for your local area network and test the communication with the ethernet controller.

4.1 Network Information

The Licel Multispectral Lidar Detector is shipped with a default static IP address. The default parameters are:

IP address	10.49.234.234
network mask	255.255.255.0
gateway	
port	2055

The network parameters should be aligned according to your local network environment. Before doing this, the system administrator should be contacted. He should provide the following information:

1. Should the ethernet controller use a dynamically assigned IP address (DHCP)?
   (a) If yes, the network parameters will be set by a DHCP server residing in your LAN. Refer to the subsection DHCP Mode (4.3.2) to enable the Licel Multispectral Lidar Detector to automatically receive the network parameters from the DHCP server.
   (b) If a static address configuration is to be used,
      i. the IP address,
      ii. the network mask,
      iii. and the gateway

   should be set by yourself. Refer to the subsection Fixed IP Address (4.3.1).
2. The default ports used by the ethernet controller are 2055 and 2056. Can these ports be used?
3. Is it necessary to change the configuration of any firewall in the case you need to access the controller outside of the LAN boundaries?

4.2 Network Preparation

After having installed the Licel Windows applications or the Licel LabVIEW modules on your PC you are ready to change the network configuration parameters of the Licel Multispectral Lidar Detector according to the local network settings described in the previous section.

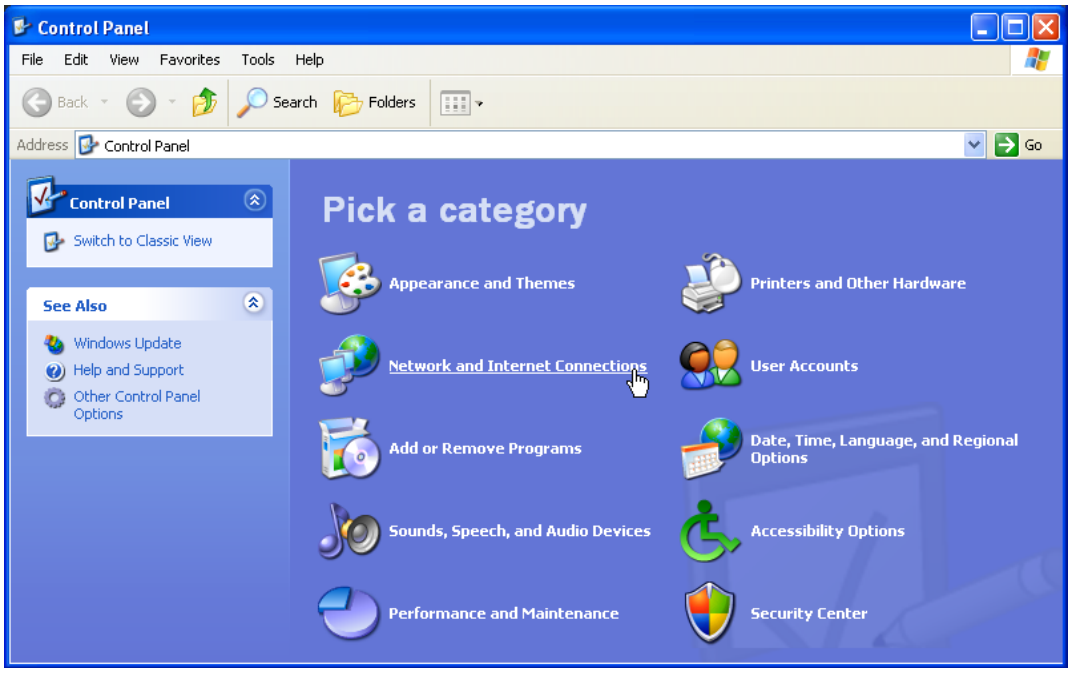
4.2.1 Establish the Connection

A straight-forward way to do this is the following procedure. You will need local administrator rights on your PC for the following steps:

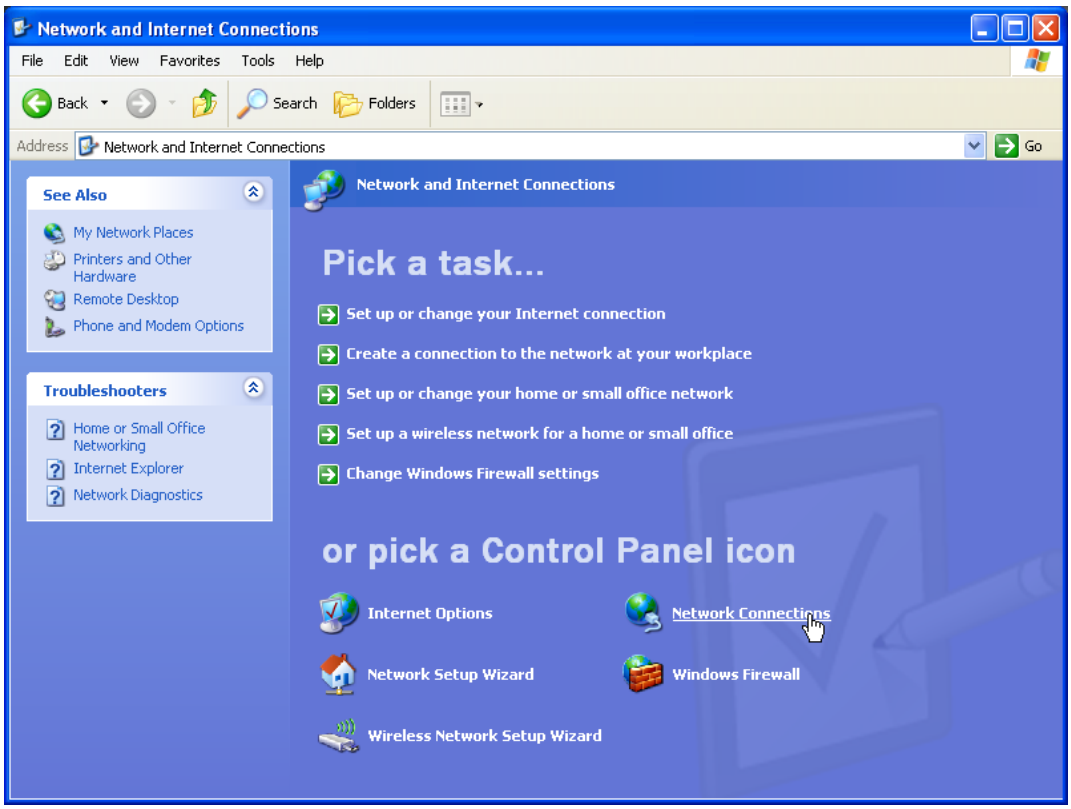
1. Disconnect the PC from the local network.
2. Open the **Properties** dialog of the network connection your ethernet adapter is assigned to. Usually you will find the appropriate network connection by opening **Network Connections** from the Windows start menu or the System Settings. The following list shows the steps to follow on a Windows XP operating system:

   (a) Click on the start button, and then on *Control Panel*.

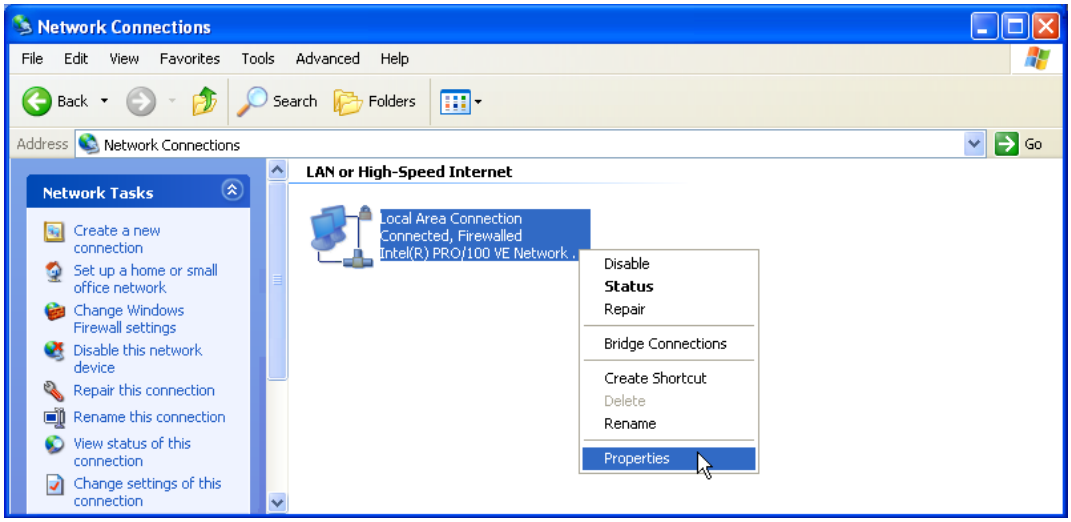
   (b) Once the control panel has come up click on *Network and Internet Connections*.



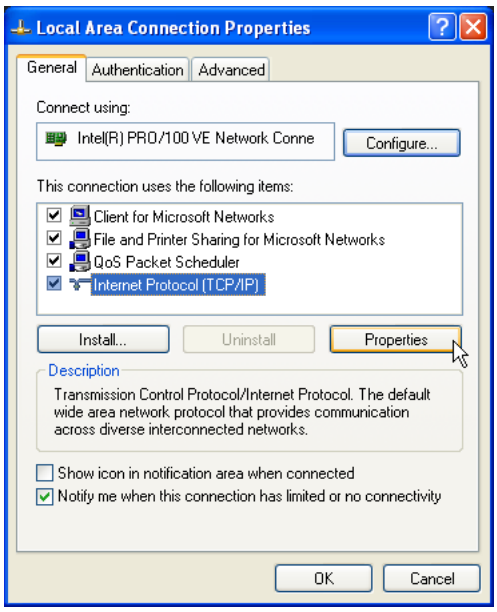
   (c) In the next window click on *Network Connections*.


Network and Internet Connections
File Edit View Favorites Tools Help
Back Search Folders
Address Network and Internet Connections
Go
See Also
My Network Places
Printers and Other Hardware
Remote Desktop
Phone and Modem Options
Troubleshooters
Home or Small Office Networking
Internet Explorer
Network Diagnostics
Network and Internet Connections
Pick a task...
Set up or change your Internet connection
Create a connection to the network at your workplace
Set up or change your home or small office network
Set up a wireless network for a home or small office
Change Windows Firewall settings
or pick a Control Panel icon
Internet Options
Network Connections
Network Setup Wizard
Windows Firewall
Wireless Network Setup Wizard

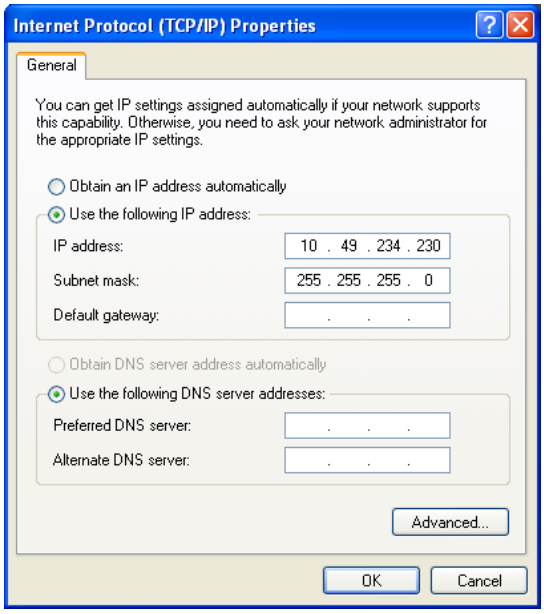
(d) The installed network connections will be shown, right-click on the local ethernet connection to be used with the Licel Multispectral Lidar Detector and choose **Properties** from the context menu.



3. Click on the TCP/IP protocol entry in the lists of components used by the assigned ethernet adapter card / LAN connection and press the *Properties* button.



4. Write down your current TCP/IP settings i.e. all settings seen in the following graphics. You will need this information to reconfigure your PC to access the LAN again.


Internet Protocol (TCP/IP) Properties
General
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.
Obtain an IP address automatically
Use the following IP address:
IP address:
10 . 49 . 234 . 230
Subnet mask:
255 . 255 . 255 . 0
Default gateway:
Obtain DNS server address automatically
Use the following DNS server addresses:
Preferred DNS server:
Alternate DNS server:
Advanced...
OK
Cancel

5. If activated, disable DHCP (checkbox *Obtain an IP address automatically*) and manually assign an IP address within the default address range of the Licel Multispectral Lidar Detector. A good choice would be `10.49.234.230`. **Never use the default address (`10.49.234.234`) of the Licel Multispectral Lidar Detector as IP address for your PC.**

6. Quit the dialog by pressing *OK*.

7. Reboot your PC.

8. Power up the rack with the Licel Multispectral Lidar Detector and connect the PC with the controller using the **cross-link cable** shipped together with your hardware.

Now you should be able to access the Licel Multispectral Lidar Detector via your ethernet card. Please test this first connection with the methods given in the next section.

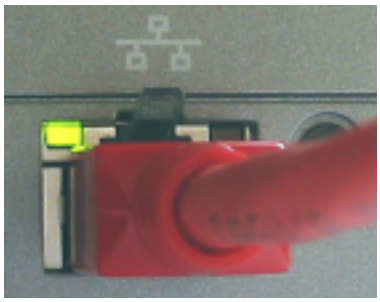
4.2.2 Diagnostics

Please carry out the following steps to verify that the connection of the Licel Multispectral Lidar Detector with the PC is established.

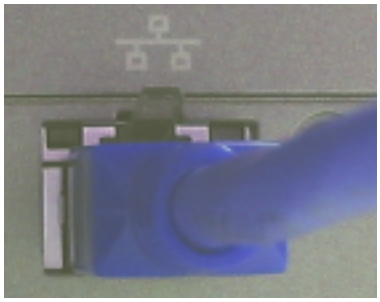
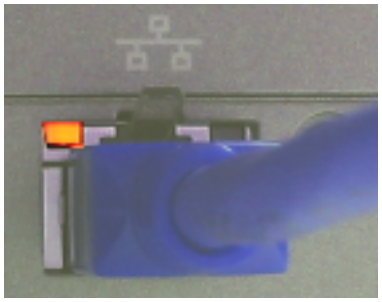
1. Verify that the green **LNK** LED lights up indicating a correct electrical connection.
2. Verify that in case of a 100Mbit ethernet connection the **Spd** lights up.
3. Verify that the network settings of your PC have changed according to your settings:
   (a) Open a command prompt window (DOS box).
   (b) Type `ipconfig` and press enter. At least one of the ethernet adapters should show the address that you previously set (`10.49.234.230`). The response should be similar to the following:

```
1 Ethernet Adapter :
        IP-Address. . . . . . . . . : 10.49.234.230
        Subnet Mask . . . . . . . . : 255.255.255.0
        Standard-Gateway. . . . . . :
```

4. Verify that the Licel Multispectral Lidar Detector is accessible via the network now:
   (a) Open a command prompt window (DOS box) or use the one from above.
   (b) Type `ping 10.49.234.234` and press enter. The Licel Multispectral Lidar Detector should respond without loss of any packet. If the controller is not responding check whether the network cable is correctly mounted and that an appropriate cable is used, i.e. a cross-link cable when working with a direct connection from the computer. Most ethernet adapters indicate a correct connection with a green LED:



A non-existent or incorrect connection is often identified by an unlighted LED (left) or red LED (right).

Please note that these indicators may be different on your PC.

   (c) If the network cable connection is correct and the controller is still not responding execute a hardware reset and repeat the procedure with the default IP address.

4.3 Network Setup

In order to configure the ethernet controller, you need either to set the controller to a fixed IP address or invoke the DHCP Mode. Whether a fixed or dynamic (DHCP) mode is used or not will depend upon your network type. Dependent on this, please refer either to the subsection Fixed IP Address or DHCP Mode and skip the corresponding other subsection. Please contact your administrator if you have not yet requested the information described in the above subsection Network Setup.

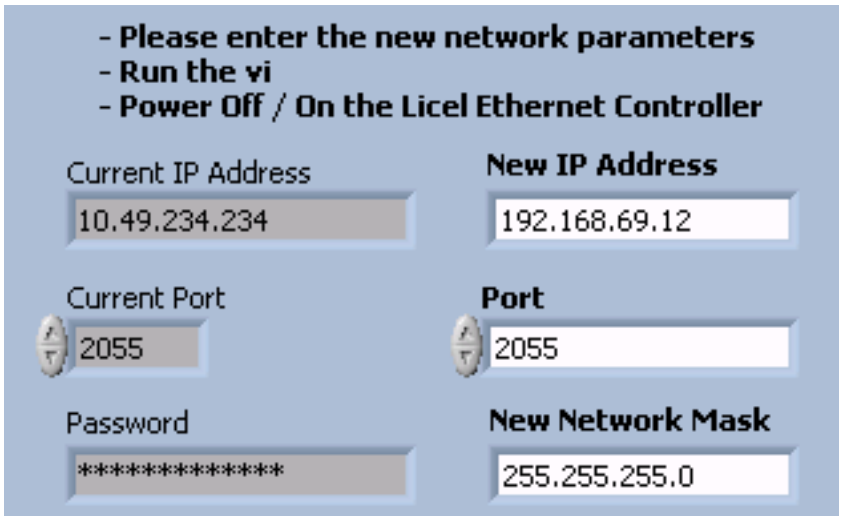
Afterwards you will have to reconfigure your PC for operating in the local network.

Once you have set the **IP Address** and **Port** for the Licel Multispectral Lidar Detector you should define these values to be used by the software.

4.3.1 Fixed IP Address

If you need to set the controller to a fixed IP address carry out the following steps. Skip the steps described in next subsection DHCP Mode.

1. Open `Licel TCPIP Set New Fixed IP Address.vi` or the corresponding Windows application from the Windows start menu.

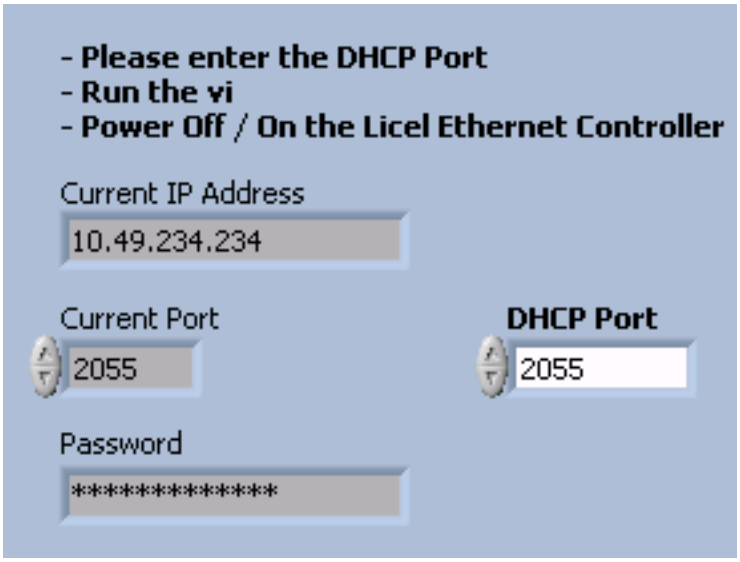


2. Set the desired network parameters in the fields **New IP Address**, **Port**, and **New Network Mask**.
3. Do not forget to enter the correct administrator **Password**.
4. Run the vi by pressing the start button. It should finish without opening an error message dialog.
5. Turn the Licel Multispectral Lidar Detector off and switch it on again. Wait **approximately 20 – 30 seconds**.
6. A `ping 10.49.234.234` executed from a command prompt (DOS box) should now time-out.

4.3.2 DHCP Mode

In order to configure the Licel Multispectral Lidar Detector for DHCP operation carry out the following steps. You must have skipped the steps described in the last subsection Fixed IP Address.

1. Open `Licel TCPIP Activate DHCP Mode.vi` or the corresponding Windows application from the Windows start menu.

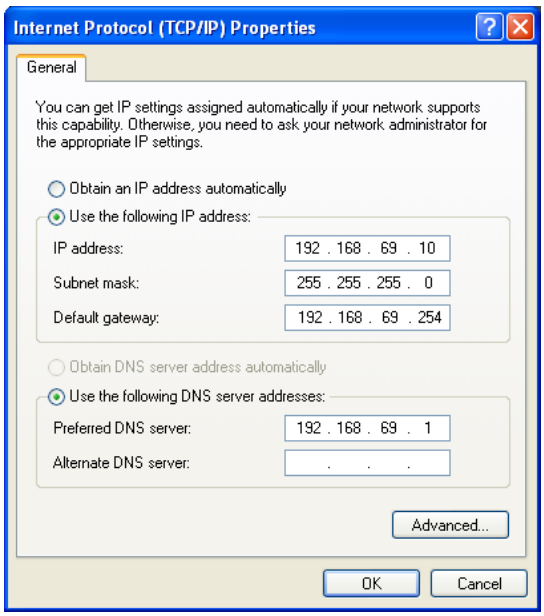


2. Set the desired **DHCP Port** number.
3. Do not forget to enter the administrator administrator **Password**.
4. Run the vi by pressing the start button. It should finish without opening an error message dialog.
5. Turn the Licel Multispectral Lidar Detector off and switch it on again. Wait **approximately 20 – 30 seconds**.
6. A `ping 10.49.234.234` executed from a command prompt (DOS box) should now time-out.

4.4 Reconfigure the PC

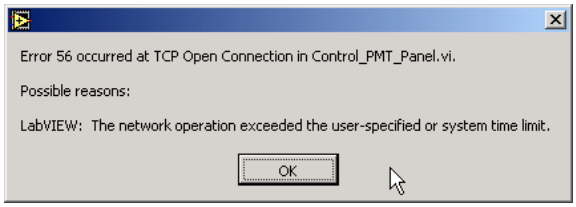
After you successfully configured the Licel Multispectral Lidar Detector the following last steps have to be carried out to reconfigure your PC for the local network and to test the connection to the Licel Multispectral Lidar Detector:

1. Reconnect the PC to the local network.
2. Open the **Properties** dialog of the network connection your ethernet adapter is assigned to. A more detailed instruction has been given above.
3. Open the **Properties** dialog of the TCP/IP protocol entry in the lists of components used by the assigned ethernet adapter card.
4. Reset your current TCP/IP settings to the values you recorded while processing the subsection to establish a network connection.


Internet Protocol (TCP/IP) Properties
General
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.
Obtain an IP address automatically
Use the following IP address:
IP address: 192 . 168 . 69 . 10
Subnet mask: 255 . 255 . 255 . 0
Default gateway: 192 . 168 . 69 . 254
Obtain DNS server address automatically
Use the following DNS server addresses:
Preferred DNS server: 192 . 168 . 69 . 1
Alternate DNS server:
Advanced...
OK
Cancel

Note that the values shown here are just example settings. You must exactly use the settings present on your PC before configuring the Licel Multispectral Lidar Detector.

5. Quit the dialog by pressing *OK*.
6. Reboot your PC.
7. Connect the Licel Multispectral Lidar Detector with your local network through a hub or switch using an **ordinary patch cable**.
8. Execute a `ping` command from a command prompt (DOS box). Use the IP address you assigned to the Licel Multispectral Lidar Detector. If the ethernet controller is in DHCP mode, you need to ask your system administrator for the assigned network address. The `ping` command's response should indicate a correctly working connection.
9. Test the access using `Licel TCPIP Getting Started.vi` or the corresponding Windows application to be started from the Windows start menu.
10. A TCP/IP timeout error with LabVIEW's error code 56 may be caused by a wrong IP address.


Error 56 occurred at TCP Open Connection in Control_PMT_Panel.vi.
Possible reasons:
LabVIEW: The network operation exceeded the user-specified or system time limit.
OK

Please check carefully that the values for **IP Address** and **Port** match with the parameters set at the Licel Multispectral Lidar Detector. Set the correct values as defaults for future operation. Other reasons for errors with code 56 are non-existing connections (check if the cable in use is correct) or unstable network operation.

4.5 TCP/IP Connection Parameters

To work properly with the Licel Multispectral Lidar Detector both the Windows applications and the LabVIEW software must be able to establish a TCP/IP connection. The user of the software must

define the **IP Address** and **Port** – these values must be equal to the parameters that have been for the Licel Multispectral Lidar Detector following the network setup section.

Defining the **IP Address** and **Port** is different for the Windows applications and the LabVIEW sources.

Windows Applications: Initialization Files

The Windows applications communicating with the Licel Multispectral Lidar Detector use an initialization files named `PMT32Channel.ini` to read their TCP/IP parameters **IP Address** and **Port**.

An example of the TCP/IP relevant information is given below:

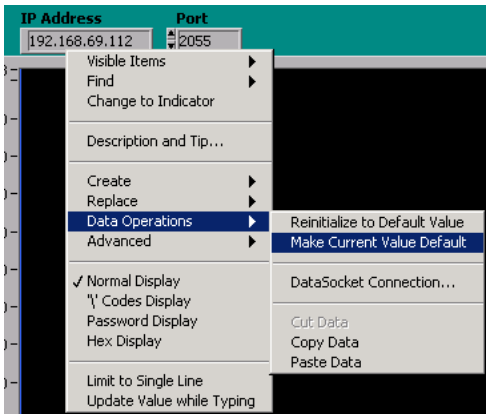
```
[TCPIP]
UseValues=TRUE
IPAddress=10.49.234.234
Port=2055
```

You may edit this file using a text editor like `Notepad` which is installed by default when setting up a Windows operating system. You may use `Notepad` as well to create a required initialization file if it does not exist in the installation directory. Make sure that you save the file before leaving the editor. You must change the values for IP address and port to the values you will set following the Instructions in the network setup section.

LabVIEW: Setting Default TCP/IP Parameters

The initialization files described above are necessary for the Windows applications because there it is not possible to set specific values as default values for control fields. However, when running the software within a LabVIEW development environment, default values can be defined for controls on the panel of a LabVIEW vi. This is especially convenient and recommended for the TCP/IP parameters **IP Address** and **Port**. Change the values to the values you set following the Instructions in the network setup section.

1. Open the vi using LabVIEW, do not run the vi.
2. Enter the value for the IP address into the control named **IP Address**.
3. Right-click on the control **IP Address** $\longrightarrow$ the context menu opens.
4. Select **Data Operations** $\longrightarrow$ a sub menu opens.
5. Select **Make Current Value Default**.

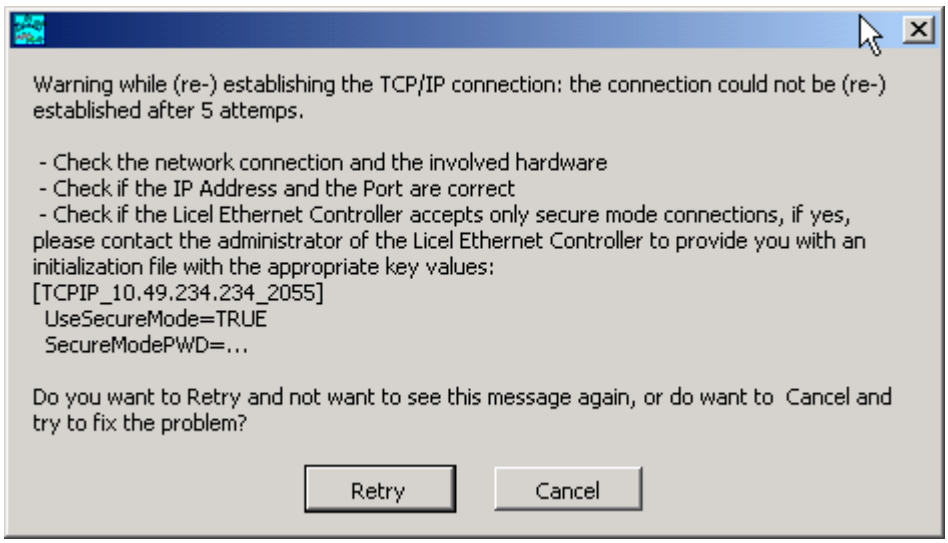


6. Repeat this procedure for **Port**.
7. Save the vi.

4.5.1 TCP/IP Connection Problems (Software)

The parent application *Licel Main Module Control* as well as the software modules when run standalone (*Track*, *Live*, *Acquis*, *Control APD-PMT*, and *Control Timing*) have a built-in mechanism to re-establish the TCP/IP connection to the Licel Multispectral Lidar Detector when the connection is lost or when the connection is not successful after the program start.

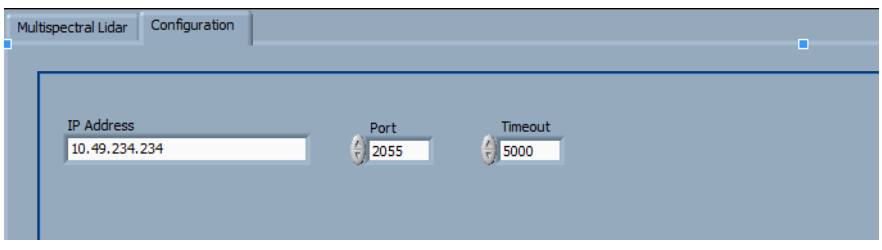
If the reconnection mechanism is not successful after 5 attempts the software assumes that some basic TCP/IP settings may be incorrect. Therefore the following error message is displayed:


Warning while (re-) establishing the TCP/IP connection: the connection could not be (re-) established after 5 attemps.
- Check the network connection and the involved hardware
- Check if the IP Address and the Port are correct
- Check if the Licel Ethernet Controller accepts only secure mode connections, if yes, please contact the administrator of the Licel Ethernet Controller to provide you with an initialization file with the appropriate key values:
[TCPIP_10.49.234.234_2055]
UseSecureMode=TRUE
SecureModePWD=...
Do you want to Retry and not want to see this message again, or do want to Cancel and try to fix the problem?
Retry
Cancel

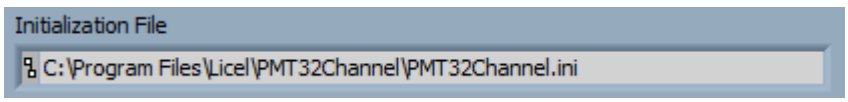
In the case that this dialog comes up please

- check the network connection and the involved hardware. Check whether the Licel Multispectral Lidar Detector and all other Licel hardware is switched on. Check that the ethernet cable is plugged correctly, and that the correct ethernet cable is used.
- check whether the **IP Address** and the **Port** the software is using equal to the values of the Licel Multispectral Lidar Detector (refer to the network setup).

  1. Before you start please enter the correct values for the **IP Address** and **Port**. You should already have set these values for the Licel Multispectral Lidar Detector
     - Using the LabVIEW vi, just enter the required values on the *TCP/IP* page and save them as defaults.



     - If you run a Windows application you should set the values in the corresponding initialization file. You will see the full path of the file in a file path indicator on the *TCP/IP* page.



While a Licel Windows application **is running** (and has not yet a TCP/IP connection) you may enter the **IP Address** and the **Port** directly. If a connection can be established (i.e. the values are correct) the parameters will be written to the appropriate initialization file when the program terminates without an error.

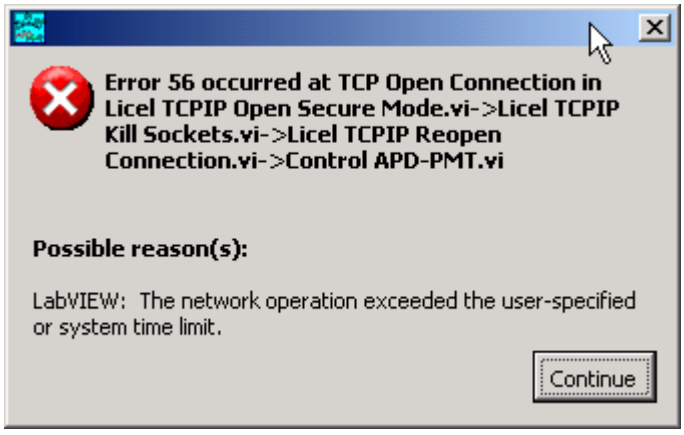
- check whether the Licel Multispectral Lidar Detector operates in secure mode. If secure mode is used please check the following section in the file `LicelTCPIP.ini`:

  ```
  [TCPIP_<Controller-IP-Address>_<Controller-Port>]
  UseSecureMode=TRUE
  SecureModePWD=<SecureModePassword>
  ```

  where `Controller-IP-Address` and `Controller-Port` are the IP address and port of the Licel Multispectral Lidar Detector, respectively. If necessary, ask your administrator for the correct password for usage in secure mode.

You have two choices to leave the message dialog:

1. Click *Retry* to continue to reconnect to the Licel Multispectral Lidar Detector.
2. Click *Cancel* to exit. The program will display an error message (here an example for *Control APD-PMT*, the dialog's appearance may depend on the LabVIEW version):


Error 56 occurred at TCP Open Connection in Licel TCPIP Open Secure Mode.vi->Licel TCPIP Kill Sockets.vi->Licel TCPIP Reopen Connection.vi->Control APD-PMT.vi
Possible reason(s):
LabVIEW: The network operation exceeded the user-specified or system time limit.
Continue

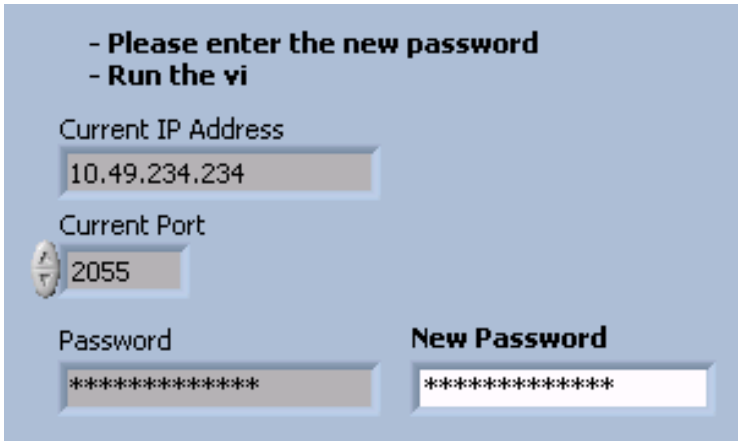
4.6 Network Security

Certain administrative tasks use an administrator password. An example is the change of the IP address of the controller. The administrator password has to be sent with the related commands.

4.6.1 Changing the Administrator Password

The Licel Multispectral Lidar Detector is shipped with the default administrator password "'*Administrator*"'. In order to change this password which grants administrative access to the controller, please carry out the following steps:

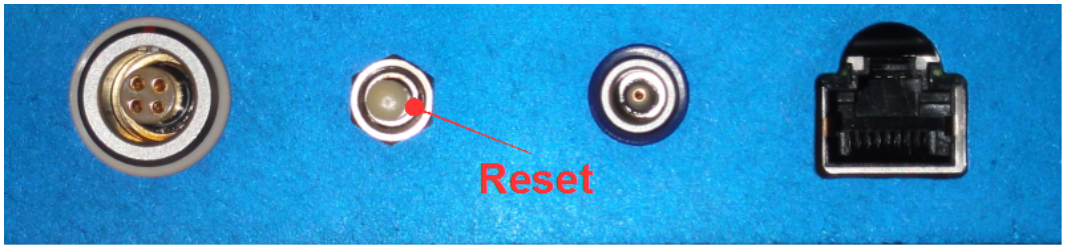
1. Open `Licel TCPIP Set New Password.vi` or start the corresponding Windows application from the Windows start menu.



2. Enter the current administrator **Password**.
3. Enter the **New Password**.
4. Run the vi by pressing the start button. It should finish without opening an error message dialog. Please note that the password is case sensitive.

4.7 Hardware Reset

A reset is performed by pressing the reset switch while powering up the controller. The reset switch is located between the DC connector and the trigger input connector.



To reset the system

- turn off the unit
- press the reset switch
- turn the rack on while keeping the switch pressed, release the switch 5 seconds after switching the unit on, wait for 30 seconds.

After a reset

- the controller has the default IP address
- the port number is reset to the default value
- the controller operates in its fixed IP address mode
- the password is reset to the default password.

Chapter 5

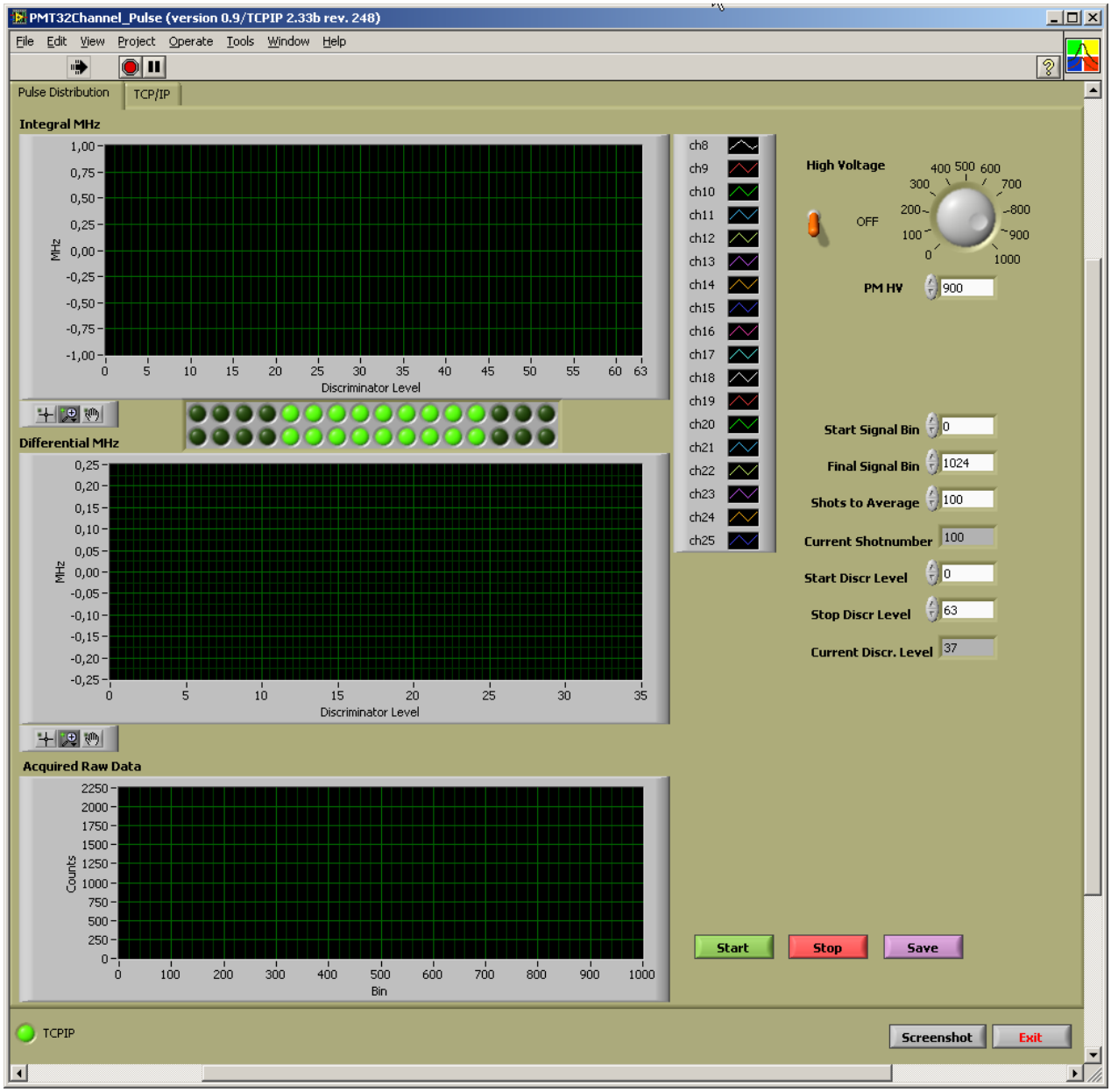
Software Modules

5.1 Overview

This chapter describes how to use the data acquisition software as well as the functions of the individual controls and indicators. In order to work through the information in this tutorial, the hardware and network setup must be completed. The next sections give brief introductions to the software module PMT32 Pulse Height Distribution, PMT32 Channel Acquisition, and PMT32 Viewer.

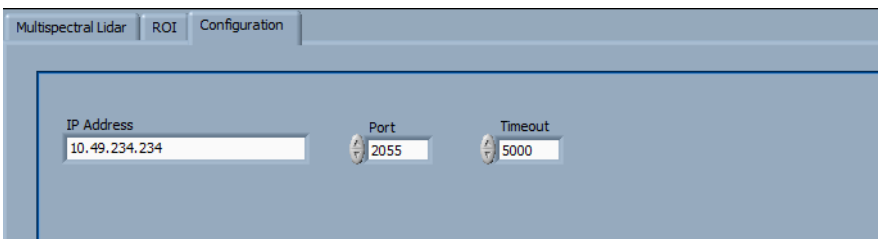
5.2 PMT32 Pulse Height Distribution

With the PMT32 Pulse Height Distribution utility you are able to acquire pulse height distributions for either all or selected channels of the Licel Multispectral Lidar Detector.
You can load the PMT32 Height Distribution utility by either double clicking on the `PMT32ChannelPulse.llb` or by selecting the virtual instrument `PMT32Channel_Pulse.vi` in `PMT32ChannelPulse.llb`. If you installed the Windows applications please start the program by selecting the entry *PMT32 Pulse Height Distribution* in the Licel section of the Windows Start menu. After doing so, you should see a screen similar to the one below.

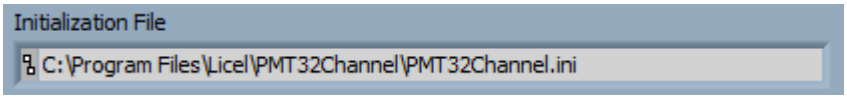


1. Before you start please enter the correct values for the **IP Address** and **Port**. You should already have set these values for the Licel Multispectral Lidar Detector.

   - Using the LabVIEW vi, just enter the required values on the *TCP/IP* page and save them as defaults.



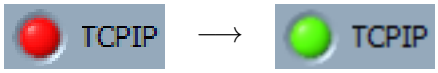
   - If you run the Windows application you should set the values in the initialization file `PMT32Channel.ini`. You will see the full path of the file in a file path indicator on the *TCP/IP* page.



2. Please switch the Licel Multispectral Detector on and connect a trigger to the trigger input.
3. To start the program press the **Run** button at the top left of the screen.

The Windows application will start automatically when called for the first time.

4. After a short time the **TCPIP** indicator should change its color from red to green indicating a successful connection with the Licel Multispectral Lidar Detector. If the indicator remains red and/or an error is indicated, please check the values for **IP Address** and **Port**, change them (on the program's panel or in the initialization file) if necessary. Check if the Licel Multispectral Lidar Detector is running and that all network connections are correct.



To record pulse height distributions please follow the steps below:

1. Make sure that your TCP/IP connection is alive,
2. Enter the desired voltage **PM HV** and switch the **High Voltage** on.
3. Select the **Start Signal Bin**, **Final Signal Bin**, and the **Shots to Average**.
4. Select the **Start Discr**iminator **Level** and the **Stop Discr**iminator **Level**.
5. Select the **Active** channel by enabling the LED type control field below the upper graphics. The numbering is as follows

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31

.

6. Click on the

button. The Licel Multispectral Lidar Detector will acquire the desired number of shots for each discriminator level between the **Start Discr**iminator **Level** and the **Stop Discr**iminator **Level**. The acquired data of the active channels is averaged between **Start Signal Bin** and **Final Signal Bin**. The result is displayed in the graphic plots **Integral MHz** and **Differential MHz**. The most bottom display is intended to show the **Acquired Raw Data** for the active channels.
7. Click on the

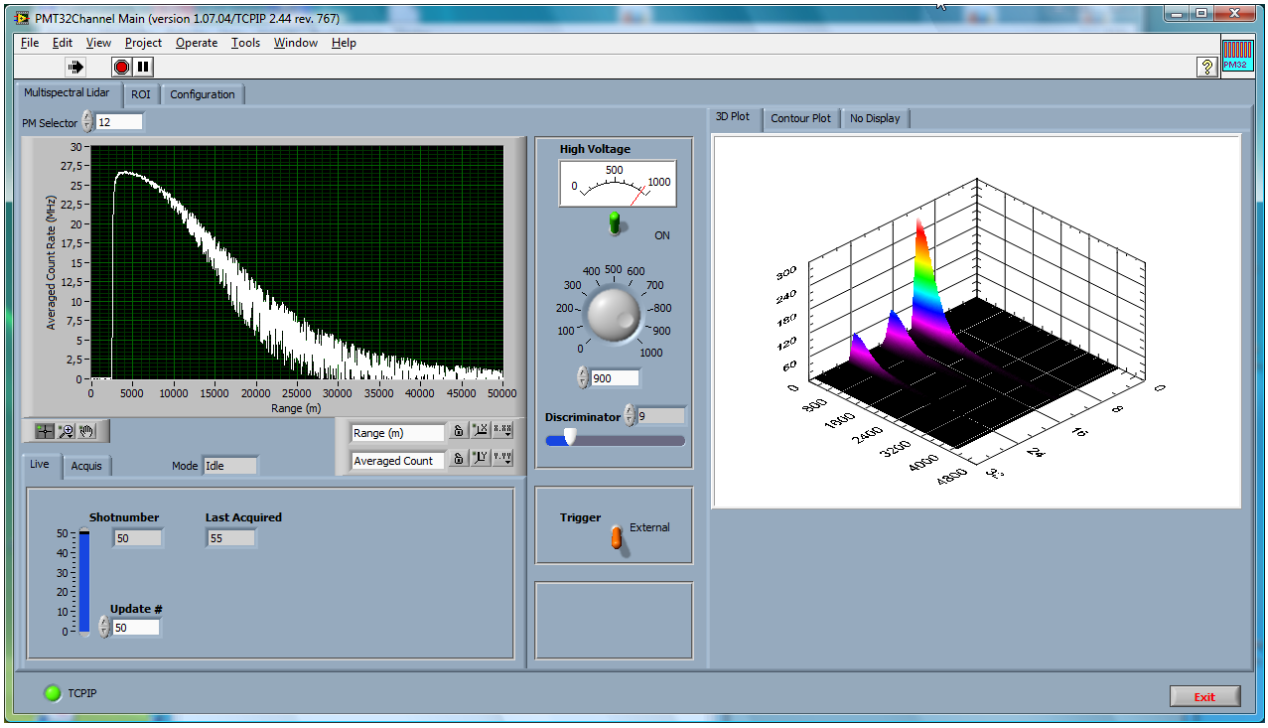
button to stop a running pulse height distribution.
8. By clicking the

button and choosing a file name in a subsequent dialog the **Inegral MHz** data is saved to a text (ascii) file.
9. Clicking the

button and choosing a file name in a subsequent dialog will save the current panel to a portable network graphics (png) file.

5.3 PMT32 Channel Acquisition

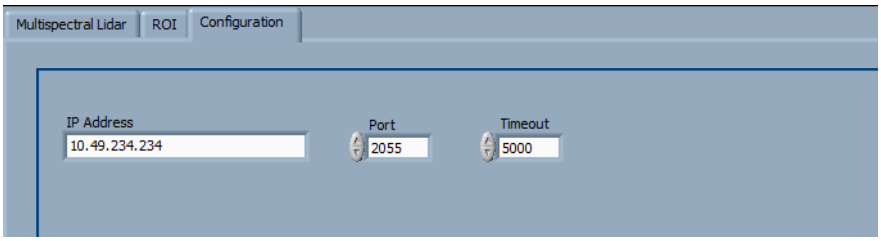
You can load PMT32 Channel Acquisition by either double clicking on the `PMT32Channel.llb` or by selecting the virtual instrument `PMT32Channel Main.vi` in `PMT32Channel.llb`. If you installed the Windows applications please start the program by selecting the entry *PMT32 Channel Acquisition* in the Licel section of the Windows Start menu. After doing so, you should see a screen similar to the one below.



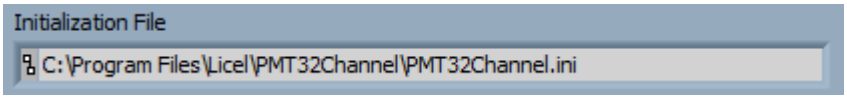
1. The program's front panel is structured using three pages, *Multispectral Lidar*, *ROI*, and *Configuration*. General settings are defined on the page *Configuration*. The acquisition settings and the main data display are located on *Multispectral Lidar*. The page *ROI* enables to inspect the 32 channel spectrum in a defineable range.

2. Before you start please enter the correct values for the **IP Address** and **Port**. You should already have set these values for the Licel Multispectral Lidar Detector.

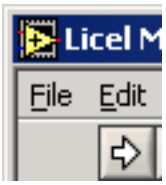
   - Using the LabVIEW vi, just enter the required values on the *TCP/IP* page and save them as defaults.



   - If you run the Windows application you should set the values in the initialization file `PMT32Channel.ini`. You will see the full path of the file in a file path indicator on the *TCP/IP* page.

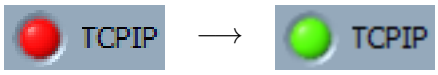


3. Please switch the Licel Multispectral Detector on and connect a trigger to the trigger input.
4. To start the program press the **Run** button at the top left of the screen.



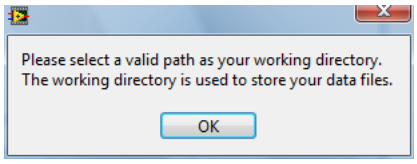
The Windows application will start automatically when called for the first time.

5. After a short time the **TCPIP** indicator should change its color from red to green indicating a successful connection with the Licel Multispectral Lidar Detector. If the indicator remains red and/or an error is indicated, please check the values for **IP Address** and **Port**, change them (on the program's panel or in the initialization file) if necessary. Check if the Licel Multispectral Lidar Detector is running and that all network connections are correct.



6. If the controller sends the data via a second TCP/IP connection a second **TCPIP** indicator will be visible.

If you run the program for the first time it is recommended to verify and set some parameters. If the storage path for the data files found in the initialization file is invalid a message will inform you:



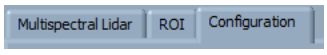
Please refer to the next subsection to set the working directory and the other parameters.

Clicking the

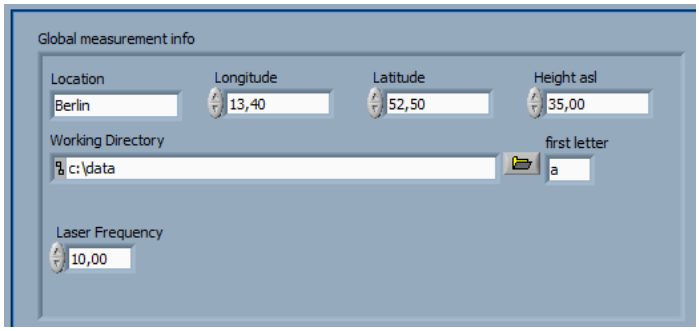
switch will stop the program.

Global Configuration Parameters

The global configuration parameters can be set in the middle of the tab page *Configuration*



:



The **Working Directory** is the location where you want data files to be stored and the **First Letter** is a letter that will be used as a prefix for the file names. Directly enter the path of the **Working Directory** into the control field or browse your file system using the browse button. The format of the file names is

```
?YYMDDhh.mmssxx
```

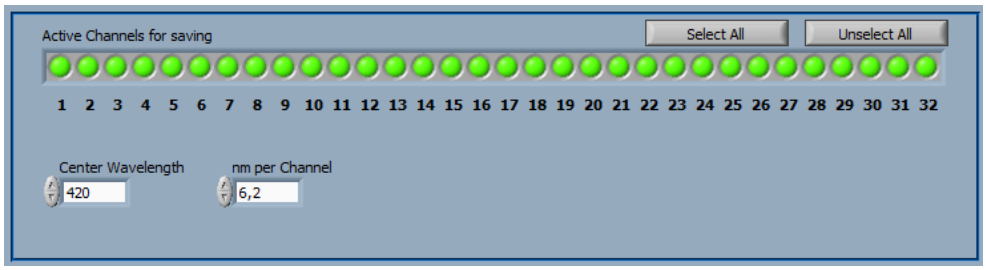
where `?` is the **First Letter**, `YY` is the year of the century, where `M` is the month (hexadecimal, 0 – C), `DD` is the day of the month, `hh` is the current hour of the day, `mm` are the minutes, `ss` the seconds, and xx the first 2 decimal places of the seconds.
For example the filename

```
a0552011.281650
```

is a file that would have been taken on May 20, 2005 (or 2105...). The operator set a to be the first letter (as in the screenshot) and the time was 11:28:16.50.
The other information above has no effect on the program execution, it is only stored in the data file headers for later reference. The fields available are your current **Location** (e.g. Berlin), the **Longitude** and **Latitude** of your location, the **Height asl** (above sea level) of the location of your acqusition system, and the repetition rate of your laser (**Laser Frequency**.
Upon starting, the program preloads the information that it finds in the file `global info.ini` that is located in the same directory as the libraries or the Windows applications, respectively. The changes you make here will be saved there when leaving the dialog by pressing **Continue**.
The data file format is described in an appendix.

Channel Configuration

Also on the *Configuration* tab page the 32 PMT channels options can be set.


Active Channels for saving
Select All
Unselect All
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Center Wavelength
420
nm per Channel
6,2

The following settings are available:

Active Channels for Saving defines the channels to be saved to data files. This setting is stored in the initialization file.

Select All selects all channels

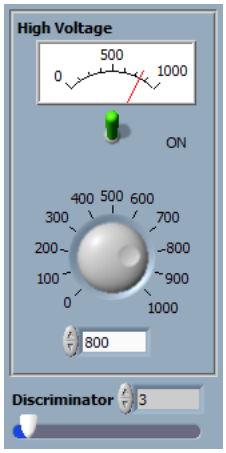
Unselect All unselects all channels

Center Wavelength this wavelength is the center wavelength of the 32 channel PM array found after a calibration. This value effects together with **nm per Channel** the wavelength of the individual channels in the header lines of the data files and the display of the spectrum on the *Multispectral Lidar* tab page. This setting is stored in the initialization file.

nm per Channel this wavelength is the center wavelength of the 32 channel PM array found after a calibration. This value effects together with **Center Wavelength** the wavelength of the individual channels in the header lines of the data files and the display of the spectrum on the *Multispectral Lidar* tab page. This setting is stored in the initialization file.

Voltage and Discriminator Level

The **High Voltage** and **Discriminator** are set in the middle of the page *Multispectral Lidar*.



Both, the **Discriminator** level and the **High Voltage** value are saved to the initialization file. The voltage set at the Licel Multispectral Lidar Detector is displayed at the top.

External / Internal Trigger

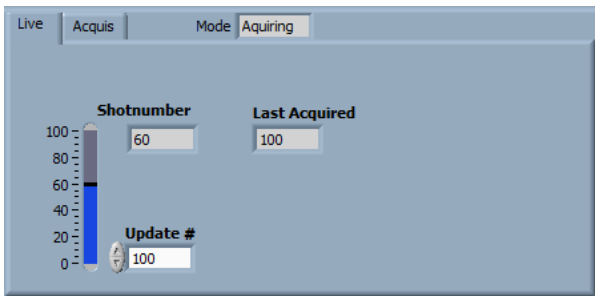
In normal operation it is assumed that an external trigger source is available. For testing purposes or calibration with a continuous spectral lamp the controller is capable to generate an internal trigger signal. The function is enabled by the **Trigger** switch.

Mechanical Shutter

A **Shutter** switch is operable if an external mechanical shutter is supported by the controller.

The Live Mode

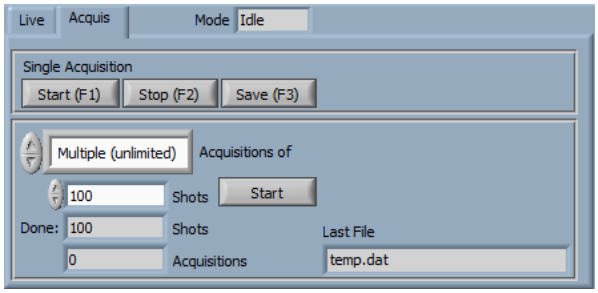
`PMT32 Channel Acquisiton` will always start in the *Live* mode. The **Mode** can be switched with the mode selector at the top in the middle of the *Multispectral Lidar* tab page. Please refer to the screenshot above.



In the oscilloscope-like *Live* mode an acquisition is started and the shot number is monitored and displayed in the **Shotnumber** field. When the detector has acquired the desired number of shots **Update #** and is ready to be read out, the data is read and displayed at the left side. The only necessary action in the software to obtain a signal is to set a value for the **High Voltage** and switch on the high voltage on the right. The **Mode** above is the current acquisition mode of the Licel Multispectral Lidar Detector.
If the **Update #** is set to 1 the acquisition is started with the `TRANSMIT` option. Data is immediately read from the Licel Multichannel Lidar Detector. If no trigger is received in this mode this is indicated at the top.

The Acquisition Mode

The *Acquisition* mode is intended for single acquisitions or a number of fixed or unlimited multiple acquisitions including file storage.



The following list explains the functionality of the buttons and input fields related to acquiring data and saving it to files.

Start (F1) starts a single acquisition with the number of **Shots** specified below. The acquisition stops when the number of **Shots** has been reached or when the user presses **Stop**. The number of acquired shots is displayed below. All acquired data are displayed and the data corresponding to the **Selected Channels for Saving** are saved to a temporary **Last File** `temp.dat` in the **Working Directory** specified before.

Stop (F2) Stops a running acquisition.

Save (F2) Saves the current temporary file to a file named according to the file name convention.

Multiple acquisitions are started with the

Start button in the bottom section from the figure above. Once a multiple acquisition has been started the label of the button changes to

Stop and could be used to stop a running acquisition. The data corresponding to a completed acquisition is written to a file in the **Working Directory**. The file name is displayed in the **Last File** field. The number of acquired shots and the number of completed acquisitions are displayed. The program acquires data and writes files until **Stop** is pressed. Optionally a fixed number of acquisitions to acquire can be set by switching the *Multiple (unlimited)* field to

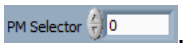
. If a fixed number of acquisitions is set the acquisition will stop automatically after the specified number of data files has been written.

Data Display

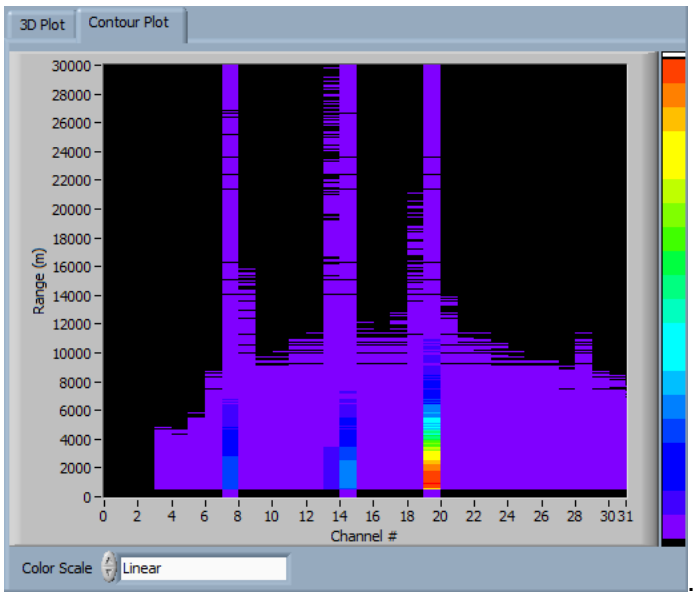
The data is displayed on the two pages *Multispectral Lidar* and *ROI*.

Multispectral Lidar

Two display indicators are available on the page *Multispectral Lidar*:

1. **Individual channel display (left)**: here the averaged count rate for an individual channel is displayed. The channel is selected by the **PM Selector** 

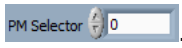
. The channel selector as well as the scale settings are used for the channel display on the page *ROI*, as well.
2. **Multidimensional Display (right)** Furthermore the data is displayed either as a *3D Plot* (see above) or a *Contour Plot*.



For the *Contour Plot* three **Color Scale** options are available, *Linear*, *Logarithmoc*, and *Auto* where all color regions represent the same number of counts.

ROI

Two display indicators are available on the page *ROI*:

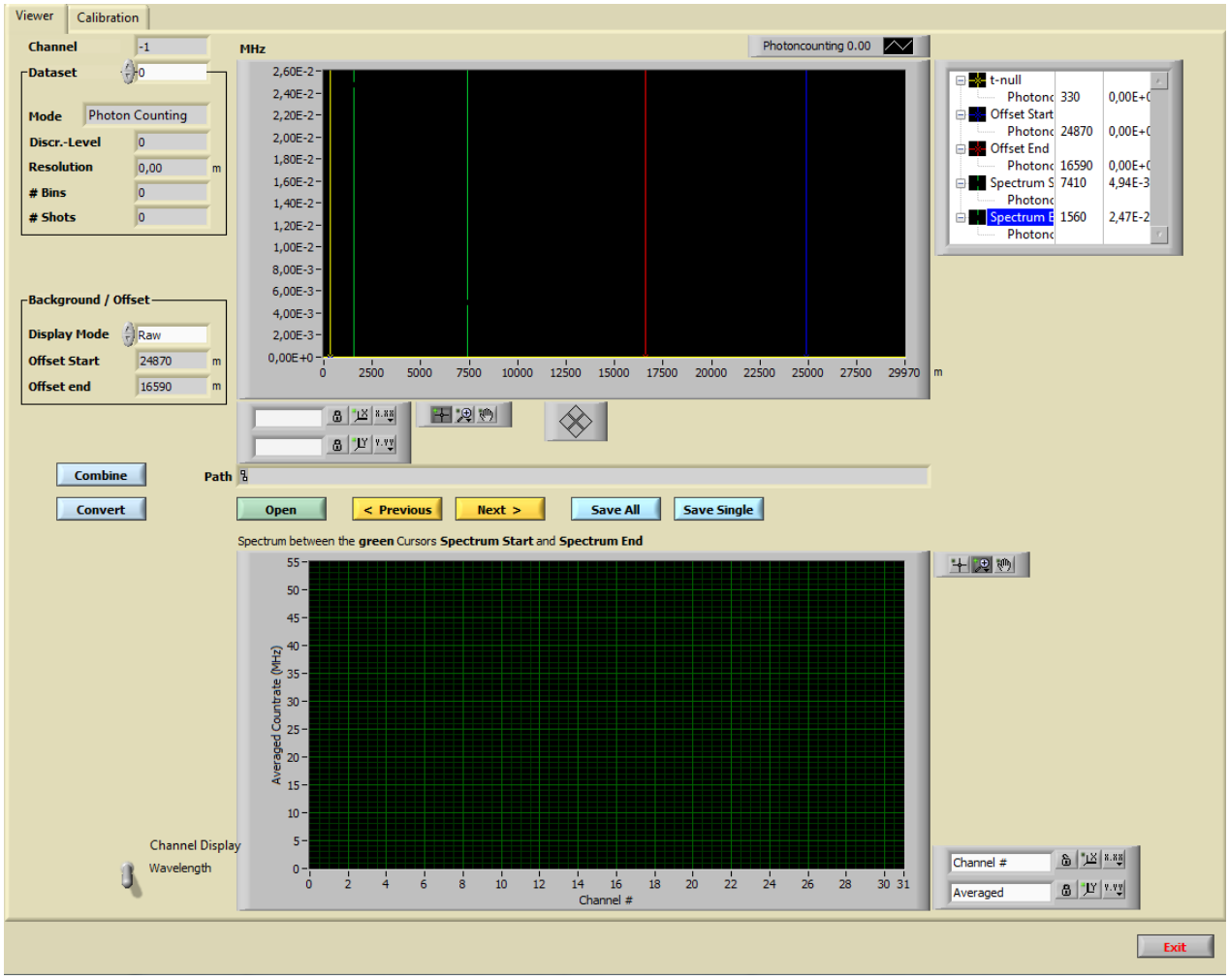
1. **Individual channel display (left)**: here the averaged count rate for an individual channel is displayed. The channel is selected by the **PM Selector** 

. The cursors are used to specify the integration region for the spectrum (right). The channel selector as well as the scale settings are used for the channel display on the page *Multispectral Lidar*, as well.

2. **Spectrum between Cursors (rigth)**: the spectrum is obtained by integrating all acquired channels in the range specified by the cursors in the individual channel display. The specified range is shown at the bottom below the graphic display. The **Channel Display** x-axis may be switched either to the channel *Index* display or to a *Wavelength* scale according to the **Center Wavelength** and **nm per Channel** from the configuration.

5.4 PMT32 Viewer

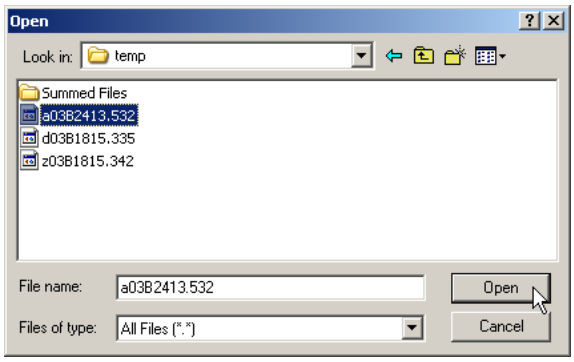
You can load the PMT32 Viewer by either double clicking on the `PMT32 Viewer.llb` or by selecting the virtual instrument `PMT32Channel Viewer.vi` in `PMT32 Viewer.llb`. If you installed the Windows applications please start the program by selecting the entry *PMT32 Viewer* in the Licel section of the Windows Start menu. After doing so, you should see a screen similar to the one below.



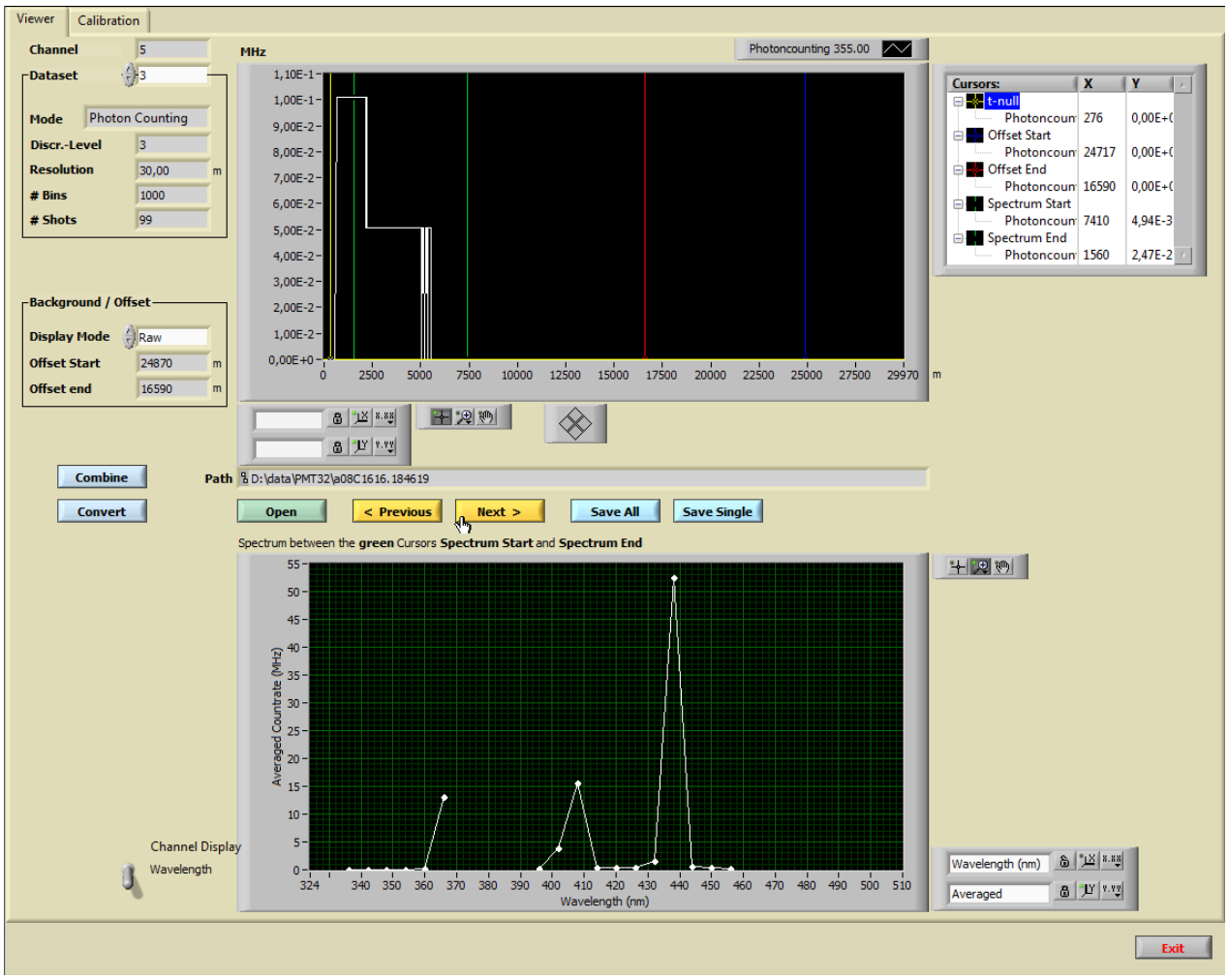
To start the program press the **Run** button at the top left of the screen.

The Windows application will start automatically when called for the first time.
A desired data file may be loaded by pressing the **Open** button.

A file selection dialog will appear. At the program start this dialog will come up without pressing the button.

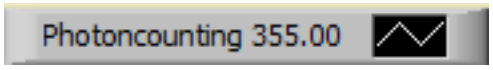


Select a valid data file, press open and the dataset number specified by the **dataset** control will appear in the graph indicator.



The channel index corresponding to the dataset read from the file is displayed top left

The wavelength is displayed in the graph legend

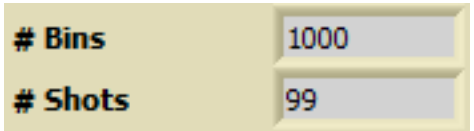


and the units used for the y-axis (MHz)is seen in the upper left hand corner of the graph

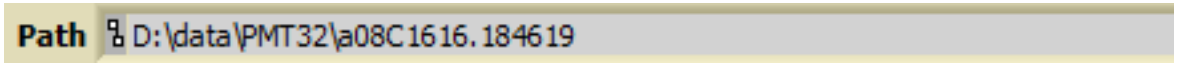
The discriminator level used during the acquisition is displayed

Furthermore the bin resolution is given in meters

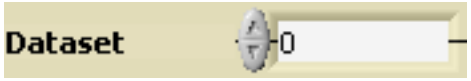
and the number of range bins and the number of acquired shots are displayed:



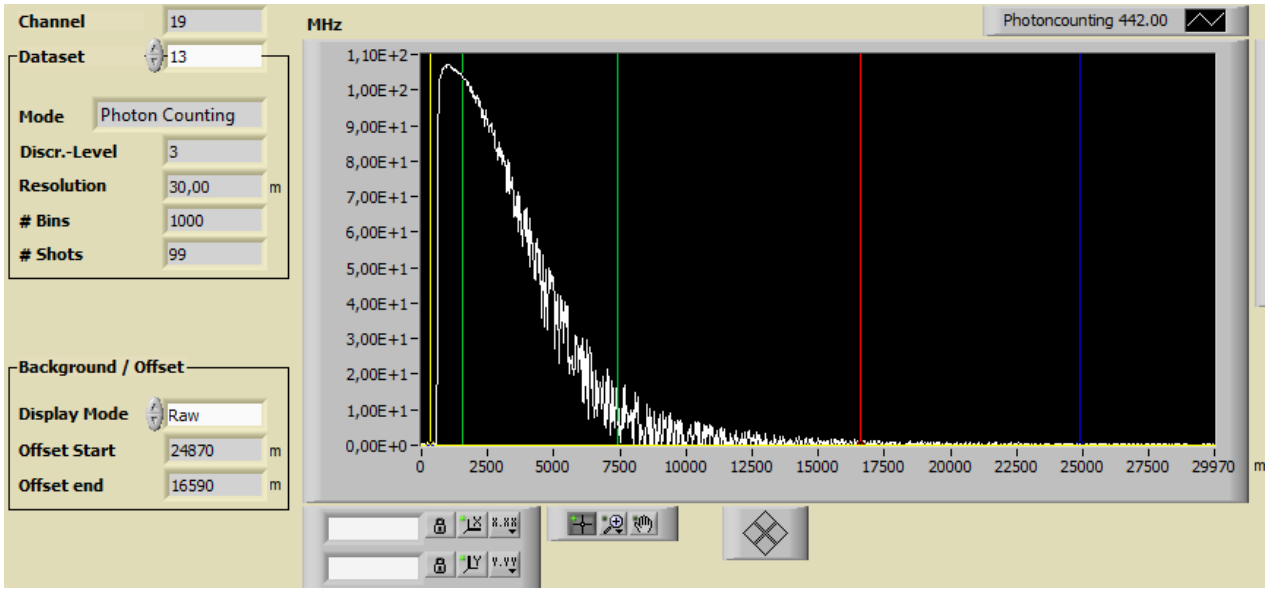
The full path to the current data file is shown in the **path** indicator.



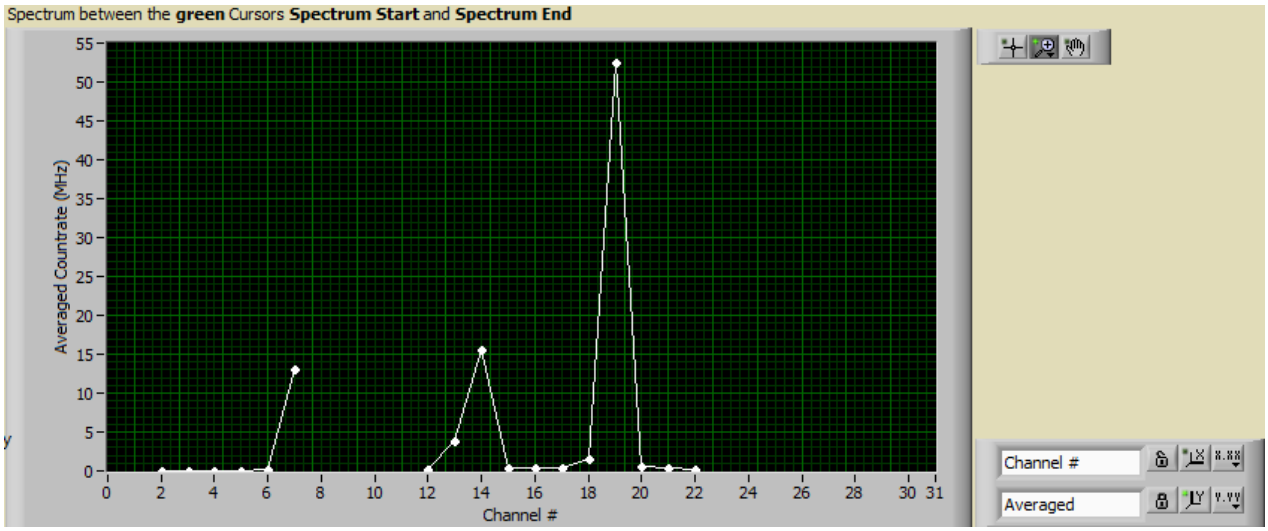
If you would like to see a different dataset (spectral channel) from the loaded file, use the **Dataset** control to choose it.



Below, dataset 13 (Channel 19 in our example) is shown.

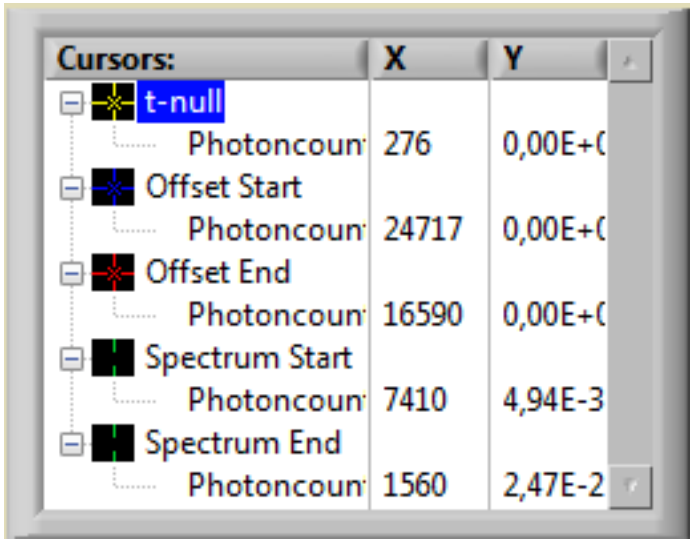


The two green cursors define the range for the integration of the spectral display at the bottom of the program's panel. Please note that not all channels have been saved in our example so the drawn line representing the spectrum is broken at the "missing" channels.

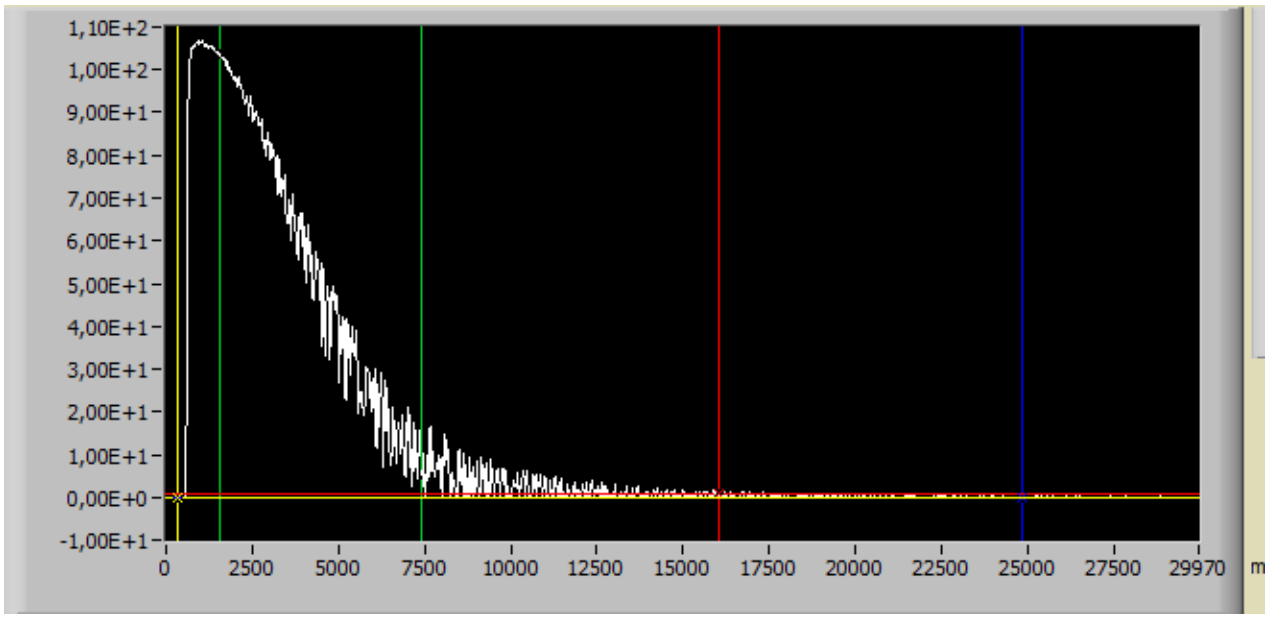


The x-axis may be set with the **Channel Display** switch either to *Index* or *Wavelength*.

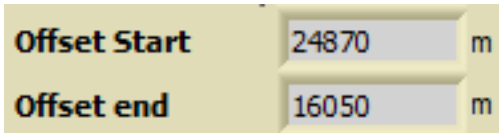
All cursor coordinates are displayed in the cursor list:



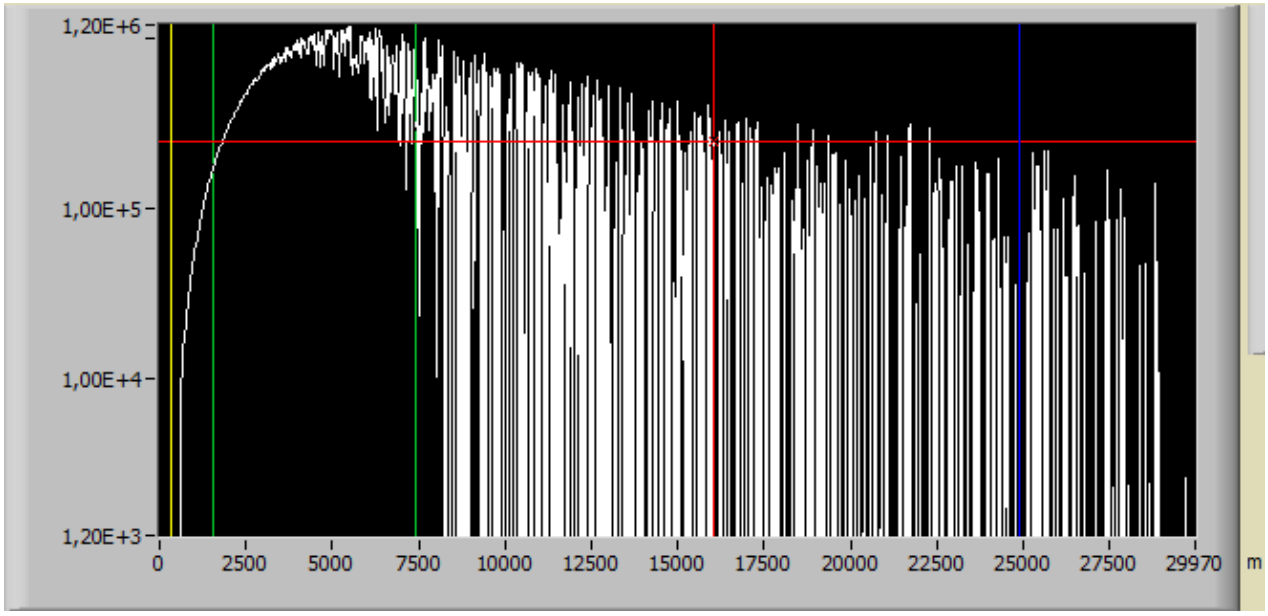
Currently the data is being shown in raw mode. The **Offset Start** (blue) and **Offset End** (red) cursors can be used to correct for the baseline offset. If the **Display Mode** *Offset Corr.* is used, then the mean value of the signal between these two cursors will be subtracted from the signal in order to create a baseline corrected signal. Use the cursor controls to move the blue and red cursors (Offset Start and Offset End) to a region which will be evaluated to generate the new baseline. Change the **Display Mode** to *Offset Corr.* and observe that the baseline of the signal changes.



In the image above, you can see that the baseline is now closer to zero. The end values of the region to be used to evaluate the baseline are shown in the **Offset Start** and **Offset End** indicators.

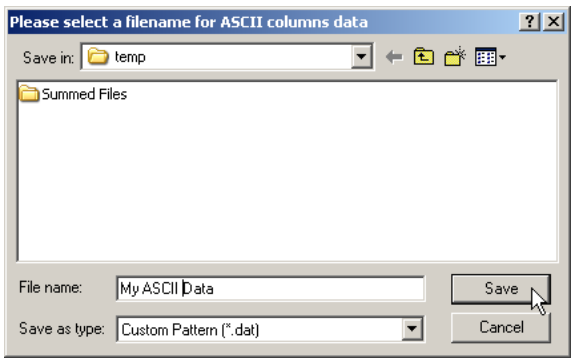


The data can also be displayed in the *Pr2* Pr^2 mode which corrects for the power loss due to the length of travel of the signal. The key parameter for the *Pr2* mode is **t-null** (yellow cursor) which defines the starting point of the signal. When switching to the *Pr2* mode, the data will look similar to the following.



Note that the signal is only corrected for power loss after the yellow cursor, which is the t-null point. All values before **t-null** are left unchanged and those after **t-null** are corrected for the power loss due to distance. The difference in the display is due to the fact that the scaling has been changed to a logarithmic scale for easier viewing.
Please remember that the *Pr2* signal is always offset corrected, as well, using the blue and red cursors.
Once you have adjusted your signal and would like to save it to a file in ASCII format as it is displayed, press the **Save Single** button.

A file dialog appears asking you to name the ASCII file.



Enter the desired file name press **Save**. The file extension `.dat` will automatically be added to the file name unless you choose a different extension. Afterwards the data is saved in ASCII format as a column and can be imported into other programs for further evaluation.

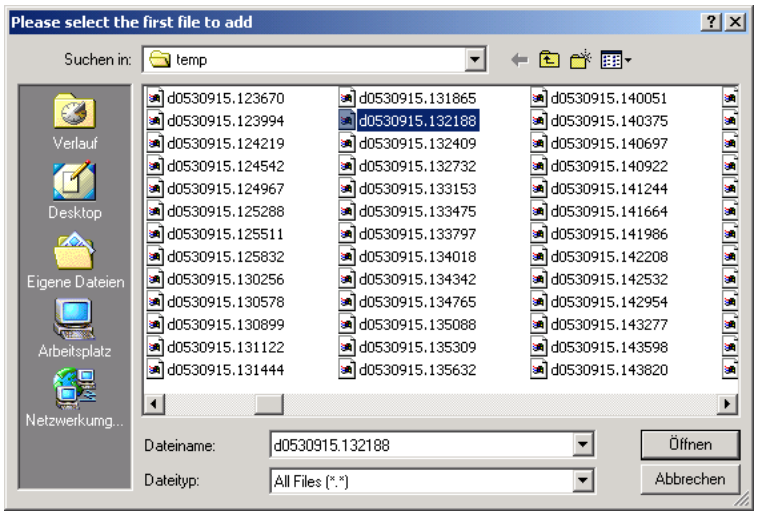
converts all datasets contained in the actual data file to an ASCII format file and appends the extension `.dat` to the end of the actual file's name. The whole file can then be imported into other programs.
If you would like to load the next file or previous file in a time series, this can be done by pressing the < **Previous** or **Next** > buttons.

By pressing one of these buttons, either the file acquired before or after the current file will be dis-

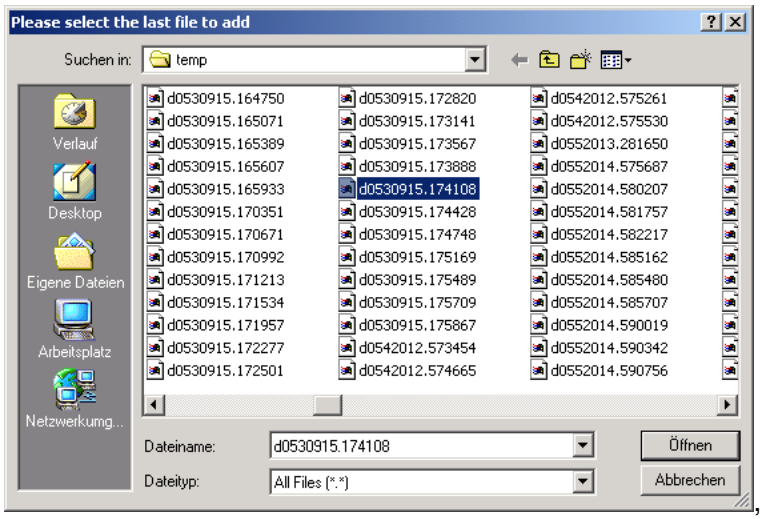
played if it exists in the same directory. You can then manipulate the data using the aforementioned capabilities of `Advanced Viewer` and save the data from the new datasets to ASCII if desired.
Two data file utilities may be called from the `Advanced Viewer`, one to sum the data values of several files to one single file (`Datafile Addfiles Interface.vi`), and another to convert the data from several files to corresponding ASCII files (`Datafile Batch Converter.vi`).
By pressing **Combine** `Datafile Addfiles Interface.vi` is interactively called to sum the data contained in a set of subsequently recorded data files.

You will have to specify:

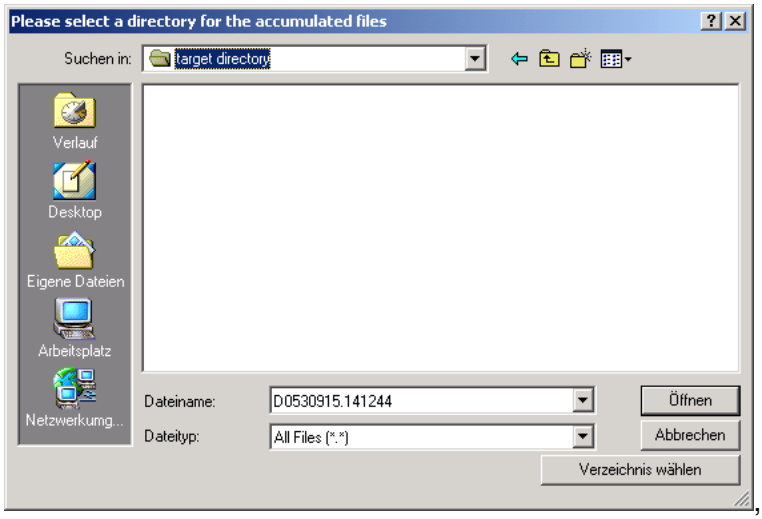
1. the name of the first file to add



2. the name of the last file to add

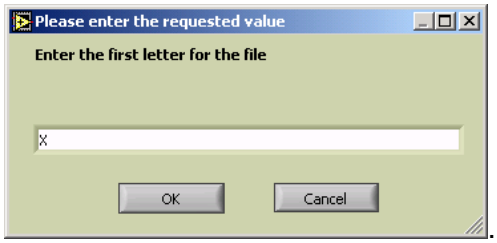


3. the name of the target directory for the file containing the summed data.



, and

4. the first letter of the file name



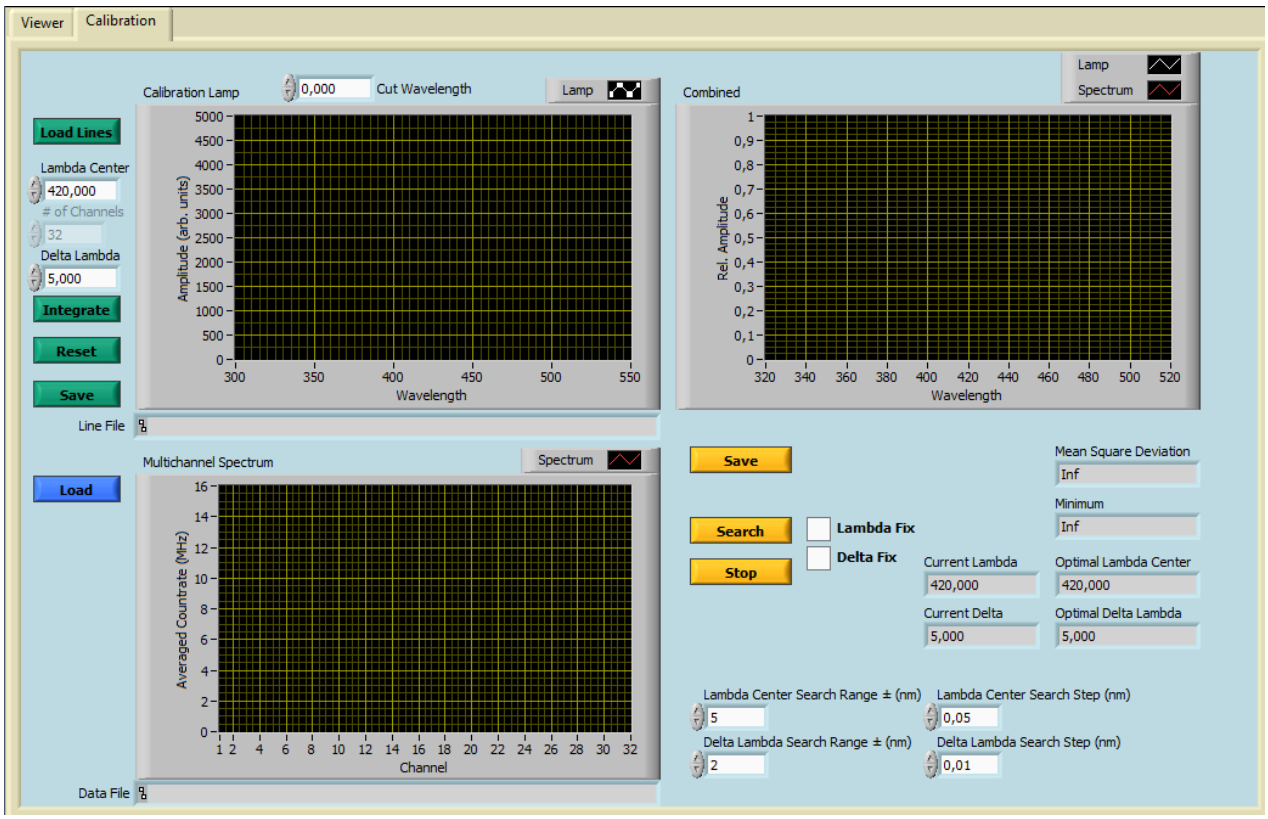
.

Both the first and the last file must reside in the same directory. The data from the files with acquisition dates/times lying between the first and the last files (including them) are summed and written to a target file into the target directory. The target file's name begins with the first letter, and the rest of the name is taken from the first selected file.
By pressing **Convert** `Datafile Batch Converter.vi` is interactively called to convert the data contained in a set of subsequently recorded data files to ASCII files.

The selection mechanism to select the first file, the last file, and the target directory is the same as for the sum operation.
Both the first and the last file must reside in the same directory. The data from the files with acquisition dates/times lying between the first and the last files (including them) are converted as described above for the **Save All** operation. Each data file will result in an ASCII file named by the original file name with the additional extension `.txt`.
The `Advanced Viewer` is terminated by pressing the **Exit Program** button.

5.5 Calibration

A calibration routine is included in the `PMT32 Viewer` to calibrate the Licel Multispectral Lidar Detector. Please note that the calibration depends on the used spectrometer and especially on the grating inside.
Please open `PMT32 Viewer` as described above and switch to the tab page *Calibration*.



The calibration is done using the following steps:

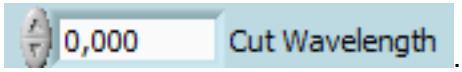
1. Acquire a spectrum of a calibration lamp (e.g. mercury) with the Licel Multichannel Lidar Detector using `PMT32Channel Main.vi` *Single Acquisition*.
2. Load a known high resolution line spectrum into the calibration routine e.g. the mercury line spectrum in the delivered file in `allLines_NIST.txt`.
3. Guess a center wavelength and a wavelength resolution (nm per channel) of the Licel Multichannel Lidar Detector. If you are using the MS125 spectrometer please refer to the spectrometer's user manual to obtain the center wavelength from the micrometer scale and the grating's number of lines per millimeter.
4. Press the *Integrate* button to integrate the high resolution spectrum using the center wavelength and the wavelength resolution. This simulates the acquisition with the Licel Multichannel Lidar Detector.
5. Compare the integrated line spectrum with the acquired spectrum and calculate the mean square deviation.
6. Change the center wavelength and / or the wavelength resolution and repeat the 2 previous steps. Minimize the mean square deviation.

This procedure is implemented in the calibration routine. The corresponding steps here are:

1. acquire a spectrum of a calibration lamp (e.g. mercury) with the Licel Multichannel Lidar Detector (*Acquisition* mode). Make sure that all 32 channels are written to the data file.
2. get a known high resolution line level spectrum for your calibration lamp. You may get spectroscopic data for many atomic transitions e.g. at http://physics.nist.gov/PhysRefData/ASD/lines_form.html. The file format needed is a text (ascii) file containing two columns, wavelength in nanometers, and the relative intensity. A line spectrum of mercury is provided in the delivered file `allLines_NIST.txt`.

(a) load the spectrum to the calibration routine using the **Load Lines** button and choosing a file in a subsequent file dialog

. The data found in the file are plotted in the to left display. You may specify a **Cut Wavelength** to simulate a wavelength high pass filter before loading the file:

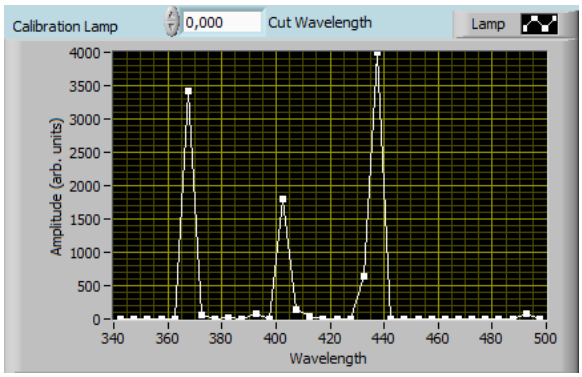


. Wavelengths below the specified **Cut Wavelength** are ignored in that case.

(b) Enter your guessed center wavelength (**Lambda Center**) and wavelength resolution (**Delta Lambda**)

. Click on the button **Integrate**

. Now the intgrated high resolution spectrum is shown in the **Calibration Lamp** display.



(c) You may reset the intgration using the button **Reset**

(d) The path of the loaded **Line File** is displayed below the graphics.

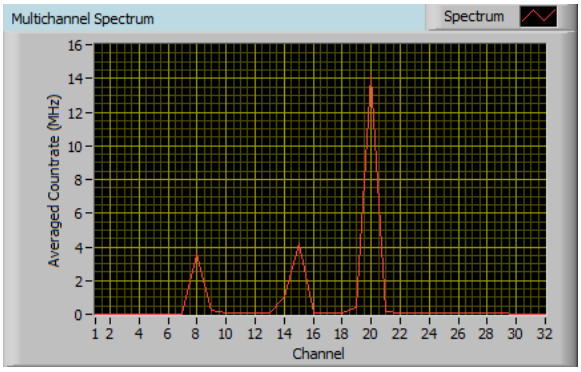
.

(e) You may save the intgrated line spectrum using the button **Save** and specifying a file name in a subsequent dialog.

.

3. (a) Load the acquired spectrum to the calibration routine using the button `Load` on the left and choosing a file in a subsequent file dialog

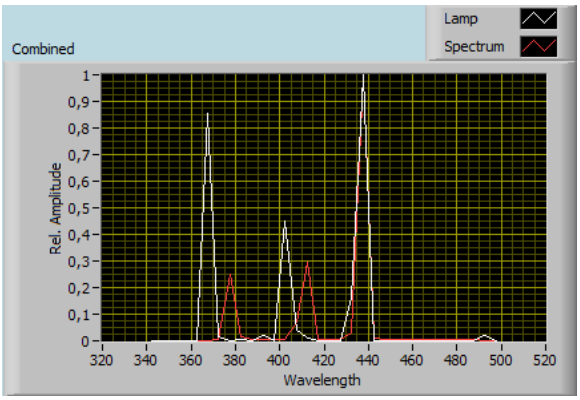
. The acquired **Multichannel Spectrum** is displayed at the bottom left.



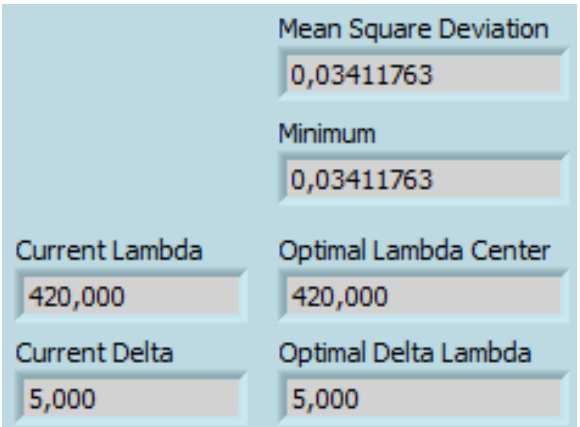
(b) The path of the loaded **Data File** is displayed below the graphics.

(c) a further click on **Integrate**

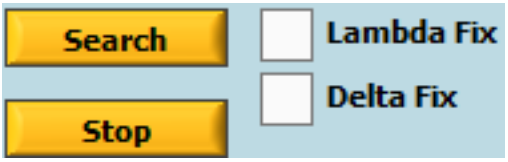
will display the acquired spectrum together with the integrated line spectrum at the top right.



4. The result of the current comparison of the integrated line spectrum with the acquired data is displayed in the corresponding indicators:



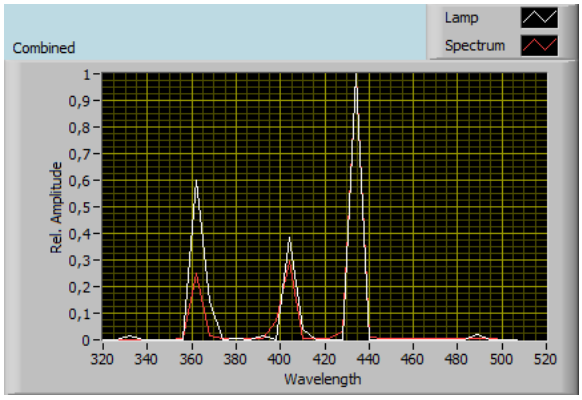
5. (a) A search will calculate the **Mean Square Deviation** after integrating the loaded line spectrum with all combinations of **Lambda Center** $\pm$ **Lambda Center Search Range** with **Delta Lambda** $\pm$ **Delta Lambda Search Range** using the step widths **Lambda Center Search Step** and **Delta Lambda Search Step**. A search is started by clicking the **Search** button.



(b) A running search can be stopped using the button **Stop**.

(c) **Lambda Center** or **Delta Lambda** can be fixed using the corresponding checkboxes **Lambda Fix** and **Delta Fix**.

6. After a search has finished the best combination of **Lambda Center** and **Delta Lambda** will be used to display the **Combined** result.



The resulting values for **Lambda Center** and **Delta Lambda** should be entered into the `PMT32 Channel Acquisition` program in the appropriate fields in the *Configuration* tab.

7. The result of the search can be saved to a text file (4 columns, index, wavelength, integrated lines, acquired spectrum) using the **Save** button

.

Chapter 6

Data Acquisition using the Licel SP32 Detector

6.1 Data Acquisition Overview

This section describes how to acquire data from the Licel SP32 Detector. It shows in detail the two modes of data acquisition supported by the detector - The PUSH mode and the SLAVE mode and lists the steps/procedures for operating the detector in the respective modes.

Please Note : Usage of the term "Detector" refers to the Licel Multispectral Lidar Detector which is abbreviated as "SP32", "Controller" refers to the Licel TCPIP Ethernet Controller present inside the SP32 and "Client" refers to the Data Acquisition Software running on the PC.

6.1.1 SLAVE Mode

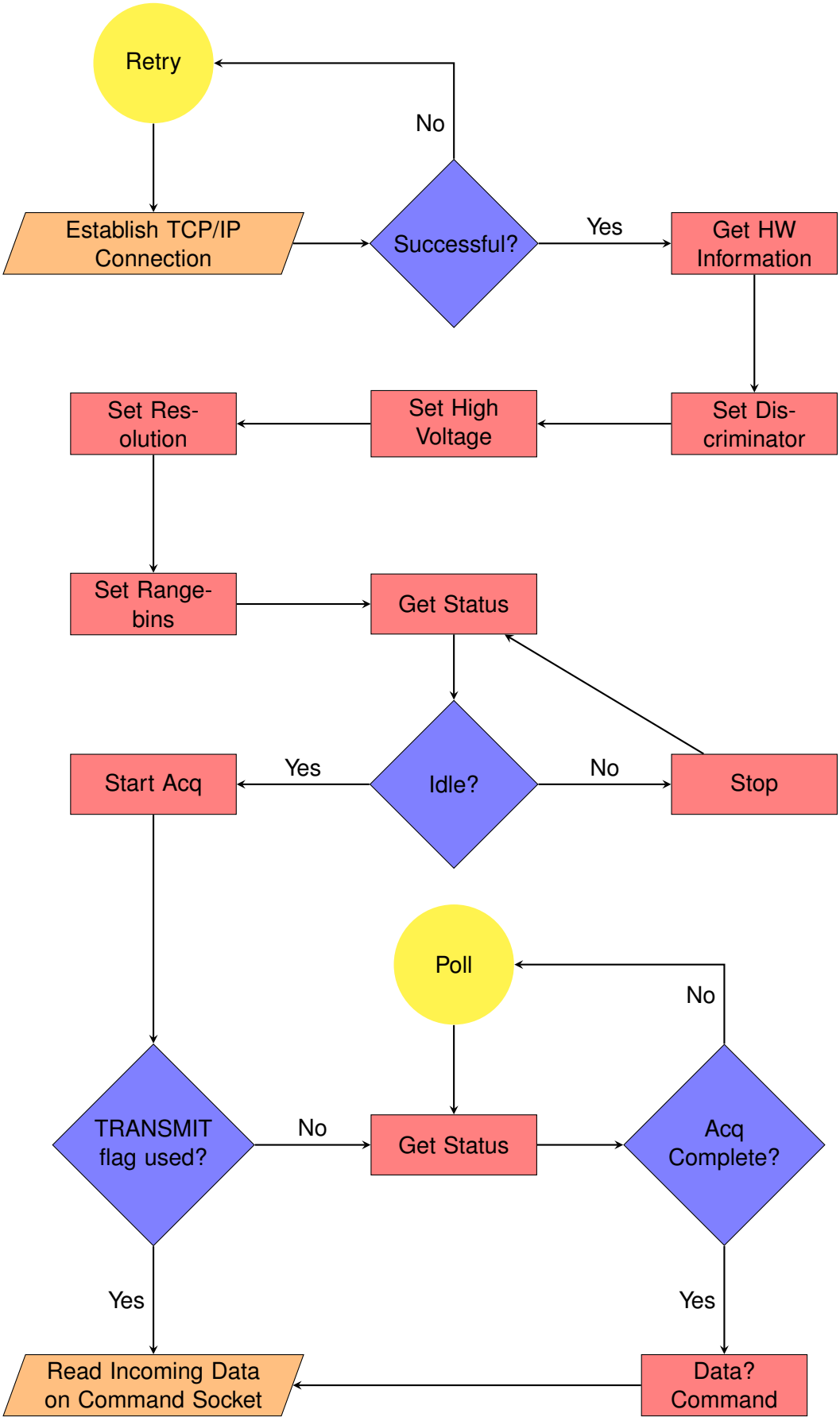
In the SLAVE mode the detector always expects a command from the client in order to perform any kind of action and send a reply. Hence in this mode detector is the SLAVE and the client is the MASTER. The different commands accepted by the detector and its corresponding replies are listed in the the TCP/IP Command List and Syntax in the appendices chapter of this manual. Please refer to the appendices for more information on the corresponding commands. The typical steps involved for acquiring data from the detector in the SLAVE mode are :

- Step 1 → TCP/IP Connection : Open up a connection to the COMMAND Socket. All the communication to the detector in the SLAVE mode takes place on the bi-directional COMMAND Socket. The COMMAND Socket is by default on port 2055.
- Step 2 → Hardware Information : Once the connection to the detector is established, the hardware information of the detector must be read out in order to know the different capabilities of the detector. This can be obtained by issuing the HW? command on the COMMAND Socket.
- Step 3 → Discriminator Level : A suitable discriminator level must be set before start of data acquisition. This can be done by issuing the DISC command on the COMMAND Socket.
- Step 4 → Highvoltage : The detector must have its Photo Multiplier Tube (PMT) high voltage turned on in order to detect photons. Set a suitable level of high voltage by issuing the PMTG command on the COMMAND Socket.
- Step 5 → Resolution : When the detector is powered on the default resolution of the detector is 10 ns. A different resolution can be set by issuing the RES command on the COMMAND Socket.
- Step 6 → Rangebins : When the detector is powered on the default rangebins of the detector is 8000. A different value can be set by issuing the RANGE command on the COMMAND Socket.

- Step 7 → Status : The current status of the detector must be determined before the start of the data acquisition. Issue a `STAT?` command on the `COMMAND Socket` to find out the current status of the detector. If the detector is not in the Idle state issue a `STOP` command on the `COMMAND Socket` to stop the detector and bring it to the idle state.
- Step 8 → Start : After all the above procedures are complete the detector can be instructed to begin its data acquisition. Issue a `START` command on the `COMMAND Socket` to start the data acquisition. The `START` command makes use of additional flags as explained in command syntax. If the detector is intended to be used in the `SLAVE` mode, care must be taken that the START command is either sent without any optional flags or with the `TRANSMIT` flag under special circumstances. Under no condition should the `PUSH` flag be used if the detector is intended to be used in the `SLAVE` mode.

  Please Note : If the `TRANSMIT` flag is used, care must be taken that no other command is sent on the `COMMAND Socket` until all the data is received from the controller. In other words, the `COMMAND Socket` must be blocked until the complete reception of data. It is for this reason that the use of the `TRANSMIT` flag is recommended only when a small `target shot number` has to be acquired.
- Step 9 → Status : If the detector was started without the `TRANSMIT` flag, the client should monitor the data acquisition status of the detector by polling with the `STAT?` command on the `COMMAND Socket`. The shot number should keep increasing till the target shots is reached. Once the target shots is reached , the data is ready to be read out. If the detector was started with the `TRANSMIT` flag , it automatically transmits the data once the acquisition is completed and hence there is no need to poll with the `STAT?` command.
- Step 10 → Data : Once the detector has acquired all the shots it is ready to transmit the data to the client. Issue the `DATA?` command on the `COMMAND Socket` to instruct the detector to transmit the data. If the `TRANSMIT` flag was used with the `START` command then the detector automatically transmits the data to the client without the need for a `DATA?` command.

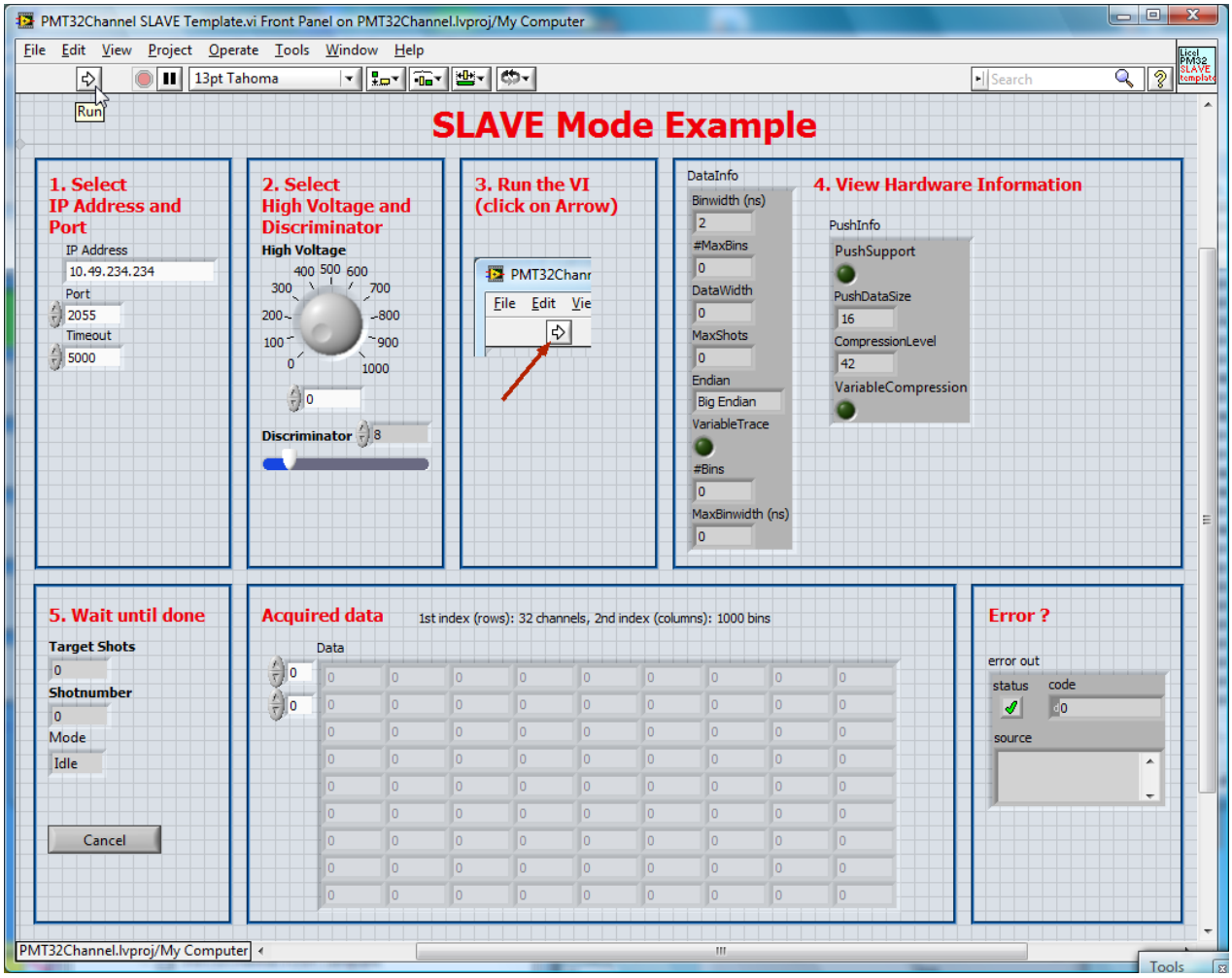
All of the above steps are illustrated in the form of a flowchart on the next page for easy reference.



Flowchart : Licel SP32 Detector SLAVE Mode Operation

LabVIEW SLAVE Mode Template

As part of the LabVIEW sources the template VI `PMT32Channel SLAVE Template.vi` is included. It is located in the LabVIEW LLB `PMT32Channel.llb` and may easily be opened from the delivered LabVIEW project.



Flowchart : Front Panel of the PMT32Channel SLAVE Template.vi

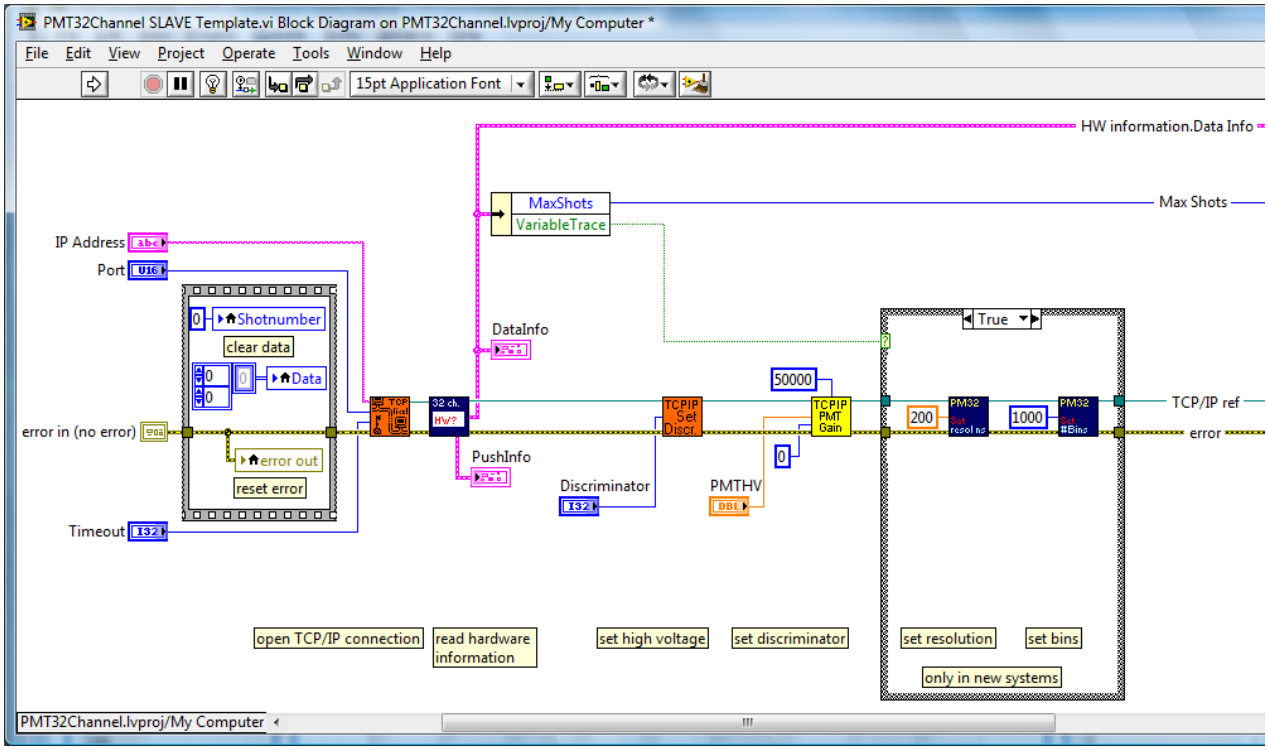
To run the template VI

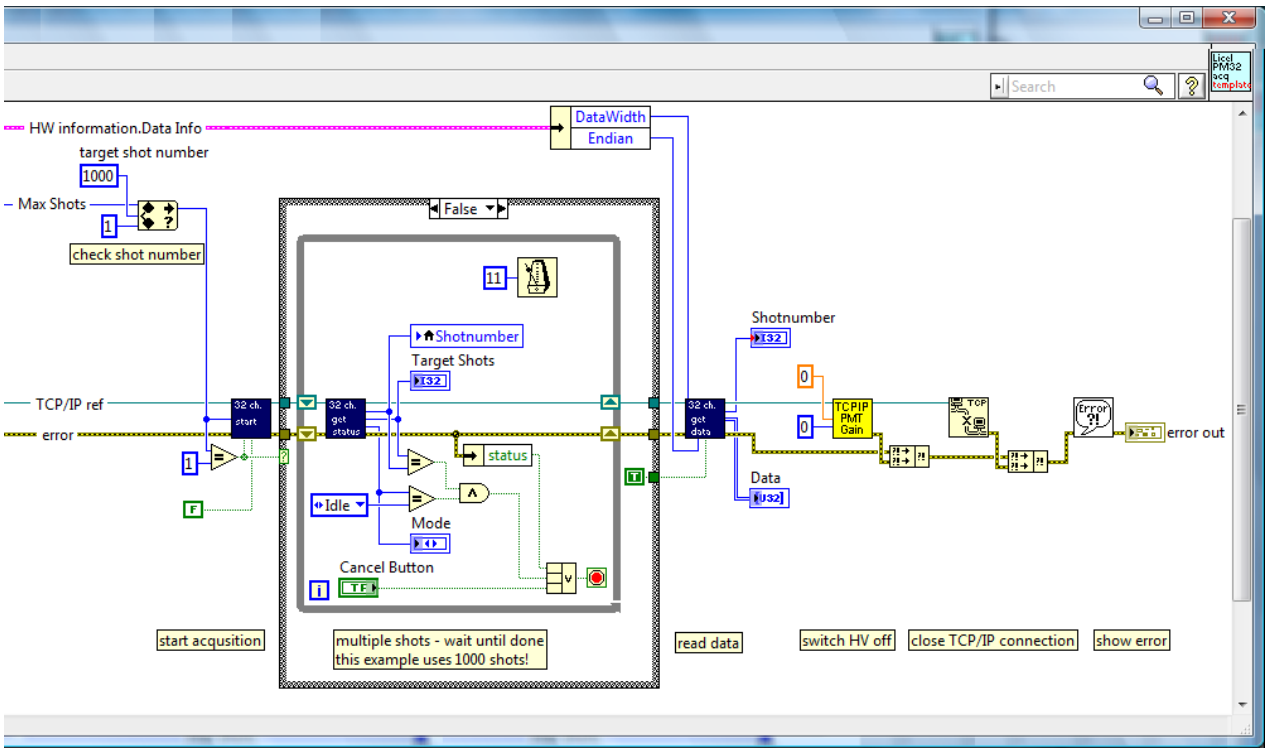
1. Select the IP Address and Port
2. Select the desired High Voltage and Discriminator level
3. Rrun the VI by clicking on the arrow icon.

The VI will then

1. Open a TCP/IP connection with the device at the given IP Address and Port (only 1 attempt in the template VI),
2. Read the hardware information and display it in the corresponding indicators,
3. Set the discriminator level to the selected value,
4. Set the high voltage to the selected value,
5. Set the resolution to 200 ns if supported by the hardware,
6. Set the range bins to 1000 if supported by the hardware,
7. Start an acquisition with 1000 shots to acquire.
8. During the acquisition the number of shots and the acquisition status is displayed,
9. When the acquisition is finished the data is read and displayed.
10. At the end the VI will set the high voltage back to 0 V
11. Close the TCP/IP connection.
12. If an error occurs it will be shown in a standard dialog and on the front panel indicator.

These execution steps can be seen on the block diagram of the VI in the next figure:


PMT32Channel SLAVE Template.vi Block Diagram on PMT32Channel.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Application Font
HW information.Data Info
MaxShots
VariableTrace
Max Shots
IP Address
Port
Shotnumber
clear data
Data
error out
reset error
error in (no error)
Timeout
DataInfo
PushInfo
Discriminator
PMTHV
50000
0
True
200
1000
TCP/IP ref
error
open TCP/IP connection
read hardware information
set high voltage
set discriminator
set resolution
set bins
only in new systems
PMT32Channel.lvproj/My Computer

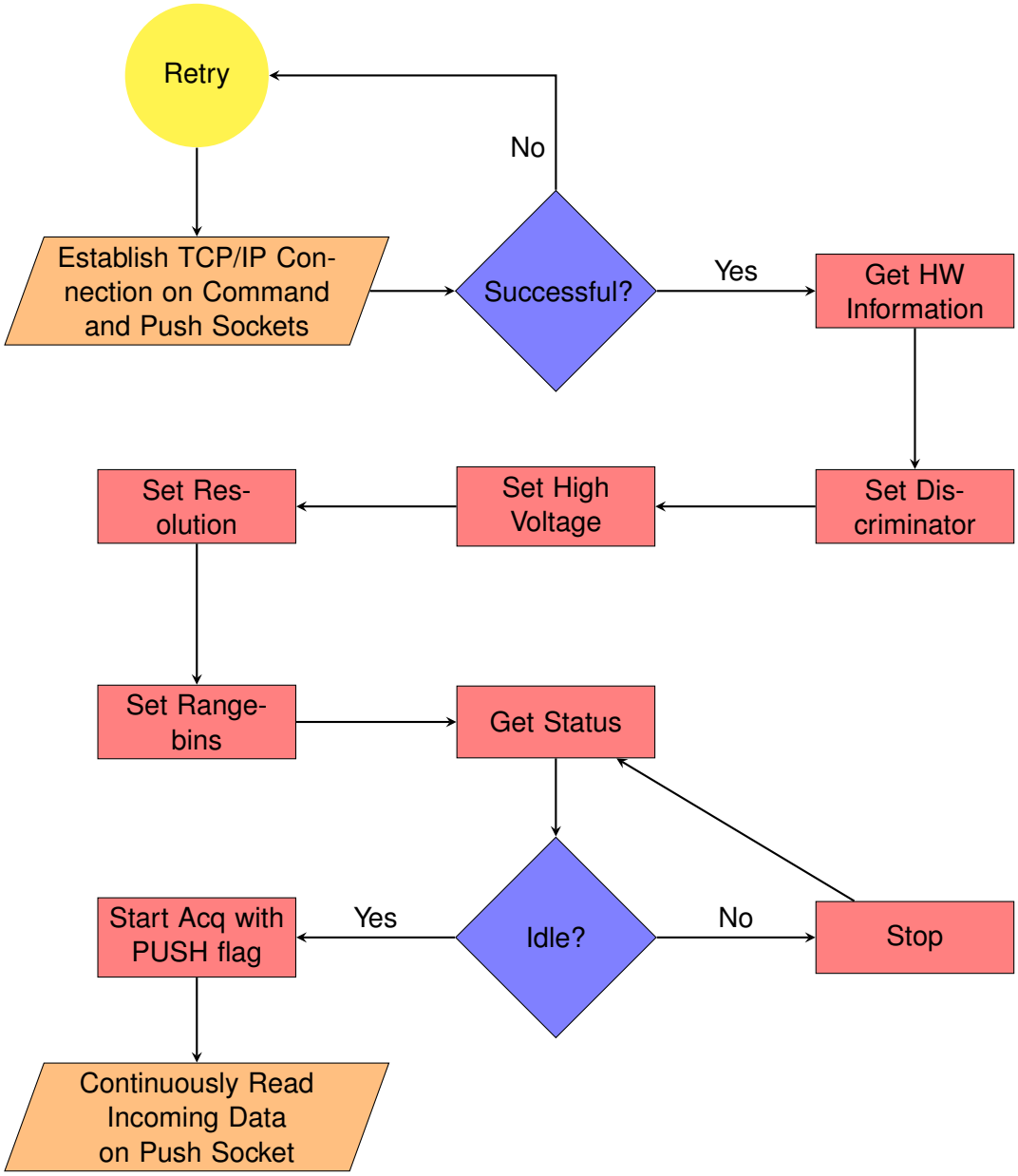

HW information.Data Info
DataWidth
Endian
target shot number
1000
Max Shots
check shot number
False
11
Shotnumber
Target Shots
TCP/IP ref
error
status
Idle
Mode
Cancel Button
Shotnumber
0
Data
error out
start acqusition
multiple shots - wait until done
this example uses 1000 shots!
read data
switch HV off
close TCP/IP connection
show error

6.1.2 PUSH Mode

The SLAVE mode can sometimes cause a lot of delays due to Nagle Algorithm when the number of shots to acquire is very low (like in the case of single shot acquisition mode) due to the large number of small sized commands being issued. In order to overcome this the PUSH mode was introduced. In this mode, the detector takes on the roll of the MASTER and the client acts as the SLAVE. The detector is by default initialized to operate in the SLAVE mode after powering on. The PUSH mode is activated only when the PUSH flag is used along with the START command. Once the PUSH mode is activated, the detector does not need any additional commands for data acquisition. It transmits the data once the predetermined number of shots are acquired and automatically restarts the detector for the next acquisition cycle. The PUSH mode makes use of a separate PUSH Socket in order to send data to the client. The PUSH Socket is on the port COMMAND Socket + 1 and hence the default port for the PUSH Socket is 2056. The PUSH Socket is a unidirectional communication socket. It is only used to send the acquisition data from the detector to the client. All commands sent to the detector should be on the COMMAND Socket and the client can not use the PUSH Socket for any purpose other than to receive the data.

Once the detector is put into the PUSH mode it starts the acquisition of the requested shots, transmits it on the PUSH Socket once the acquisition is completed and restarts itself for the acquisition of the next cycle. This process loops until a STOP command is sent by the client on the COMMAND Socket. The STOP command disables the PUSH mode and sets the detector back into the SLAVE mode. Please refer to the section on Efficient use of SP32 for Data Acquisition for more information on when to make use of the PUSH mode. The typical steps involved in acquiring data from the detector in the PUSH mode are :

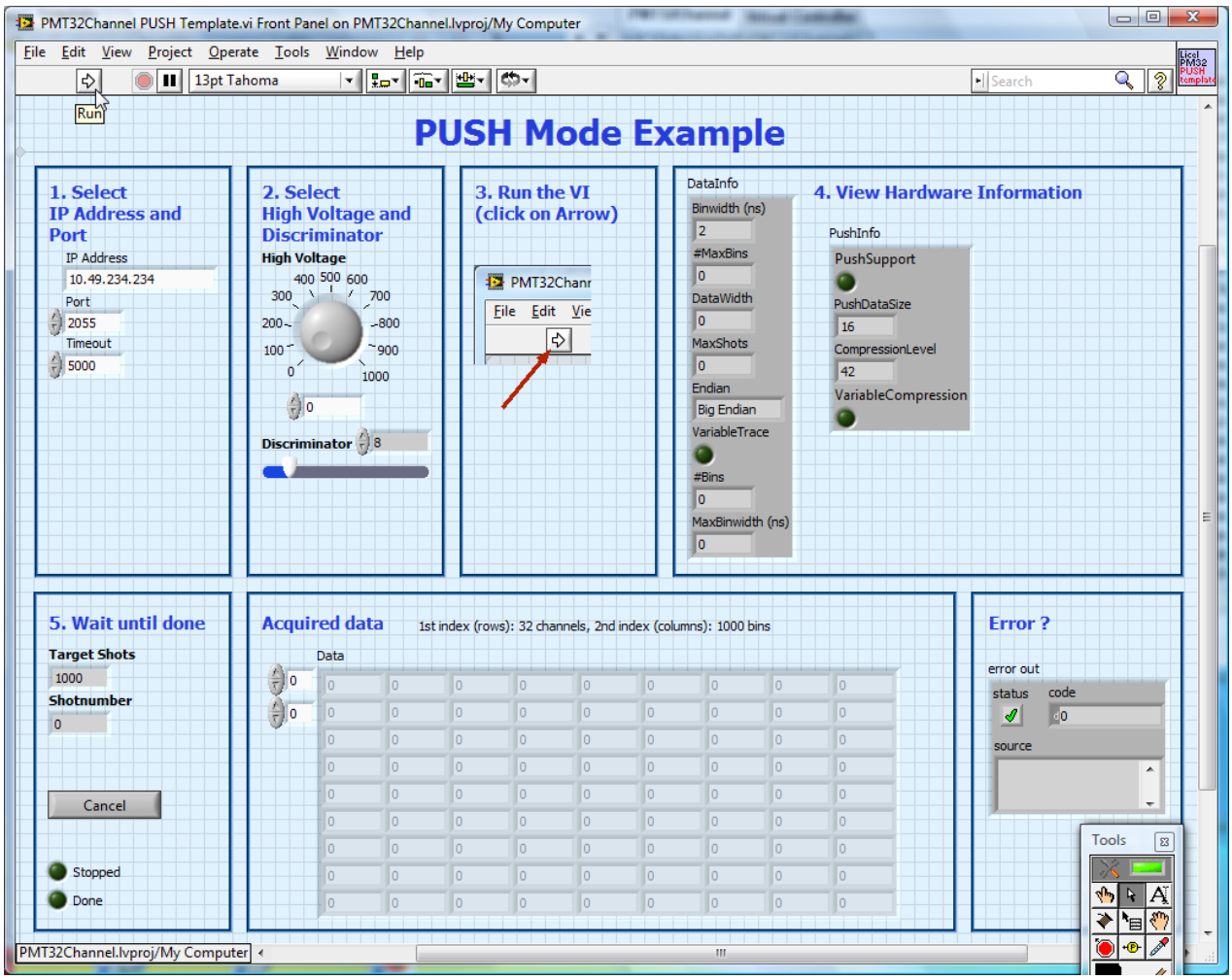
- Step 1 → TCP/IP Connection : Open up a connection to the COMMAND Socket and PUSH Socket on the corresponding ports. The detector makes use of the COMMAND Socket to receive commands and send replies. The PUSH Socket is used to send acquisition status and data to the client.
- Step 2 → Step 7 : These steps are exactly the same as in the SLAVE mode Operation.
- Step 6 → Start : Issue a START command for an acquisition of the required number of shots capable of being averaged internally along with the PUSH flag enabled. The maximum shots capable of being averaged internally is determined by the HW? command reply by the MaxPushShots parameter. The PUSH mode is only enabled after the START command is sent with the PUSH flag. Once the PUSH mode is enabled, all the client has to do is read and process the incoming data on the PUSH Socket.



Flowchart : Licel SP32 Detector PUSH Mode Operation

LabVIEW PUSH Mode Template

As part of the LabVIEW sources the template VI `PMT32Channel PUSH Template.vi` is included. It is located in the LabVIEW LLB `PMT32Channel.llb` and may easily be opened from the delivered LabVIEW project.



Flowchart : Front Panel of the PMT32Channel SLAVE Template.vi

To run the template VI

1. Select the IP Address and Port
2. Select the desired High Voltage and Discriminator level
3. Run the VI by clicking on the arrow icon.

The VI will then

1. Open a TCP/IP connection (command socket) with the device at the given IP Address and Port (only 1 attempt in the template VI),
2. Read the hardware information and display it in the corresponding indicators.
3. If the PUSH mode is not supported by the PM32 detector controller the program will terminate.

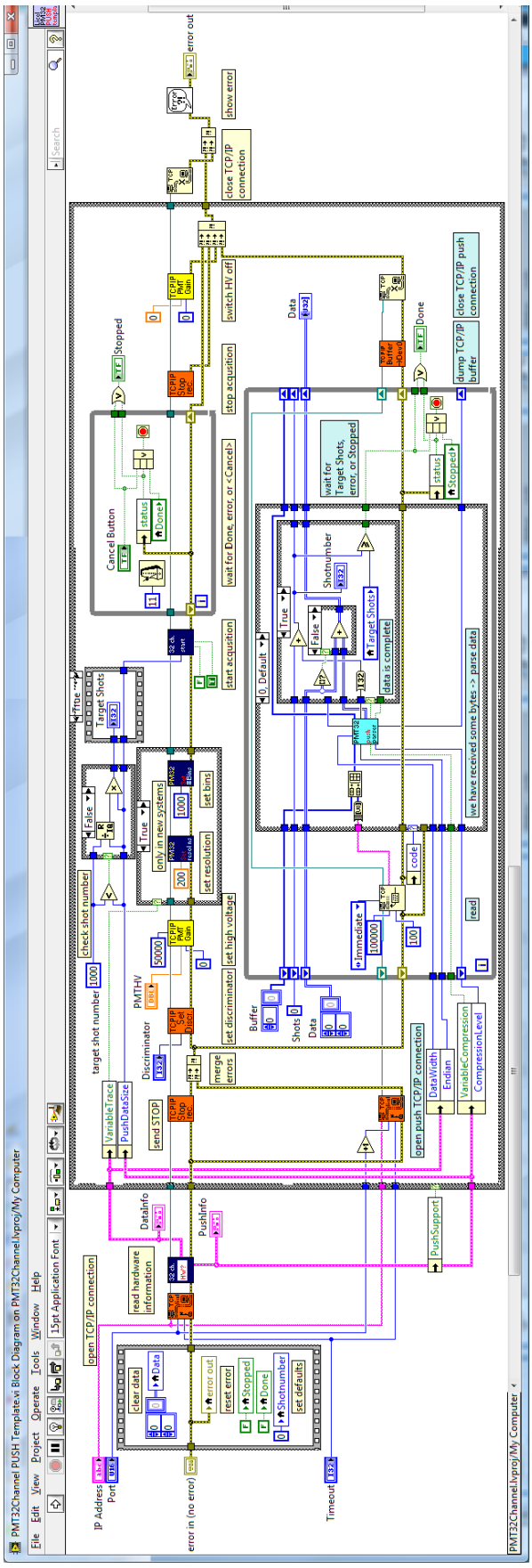
If the PUSH mode is supported the VI proceeds by executing the following operations in parallel on the `COMMAND` and `PUSH` Socket.

	COMMAND Socket	**PUSH Socket**
4.	Sending a STOP command to stop any data transfer (e.g. if the software has not been correctly stopped in a previous run)	Open a TCP/IP connection (PUSH socket) with the device at the given IP Address and (Port + 1).
5.	Set the discriminator level to the selected value,	
6.	Set the high voltage to the selected value,	
7.	Set the resolution to `200` if supported by the hardware,	
8.	Set the range bins to`1000` if supported by the hardware,	
9.	Start a PUSH mode acquisition.	
10.	Stop if the acquisition at the PUSH socket has been **Done** or <**Cancel**> is pressed ((then set **Stopped** = TRUE)) or a previous error has been detected.	Read bytes from the PUSH socket. Extract data from complete data telegrams. Update the shot number. Stop if the **Target Shots** have been reached (then set **Done** = TRUE) or the acquisition has been **Stopped** or a previous error has been detected.
11.		The acquired data is displayed.
12.	A STOP command is sent via the command socket	Data at the PUSH socket is trashed.
13.	The VI sets the high voltage to 0 V.	The TCP/IP PUSH socket is closed

Program end:

14. The will close the TCP/IP connection.
15. If an error occurs it will be shown in a standard dialog and on the front panel indicator.

These execution steps can be seen on the block diagram of the VI in the next figure:



6.2 Transmitted Data Format

The transmitted data from the detector consists of two parts namely `Preamble/Header` followed by the `Channel Trace Data`. The `Preamble/Header` contains information about the current data set necessary for the client to read the incoming `Channel Trace Data`. Both these parts have different formats based on the mode of operation of the detector.

6.2.1 SLAVE Mode

Preamble/Header

The `Preamble/Header` of the data set transmitted in the `SLAVE` mode has a fixed 16 bytes length.

Dataset Marker = 0xFFFFFFFF [4 bytes]
Acquired Shots [4 bytes]
Number of Traces [4 bytes]
Rangebins [4 bytes]

Figure : Slave Mode Transmission Data Preamble/Header

Dataset Marker [4 bytes unsigned int] :
The `Data Marker` is a fixed `4` byte value used to denote the start of a data set. The client must use the `Data Marker` of the `Preamble/Header` as a reference to the start of the data set for further processing.

Acquired Shots [4 bytes unsigned int] :
The `Acquired Shots` is a `4` byte value used to denote the number of internally averaged shots present in the current data set. This has to be equal to the `target shot number` of the `START` command.

Number of Traces [4 bytes unsigned int] :
The `Number of Traces` is a `4` byte value used to denote the total number of traces present in the acquired data. In the `SLAVE` mode each trace corresponds to a single channel of the detector. Hence this is a fixed value of `32`.

Rangebins [4 bytes unsigned int] :
The `Rangebins` is a `4` byte value used to denote the total number of rangebins present in each trace of the data set. This is equal to the `RangeBins` parameter of the `HW?` command reply.

Channel Trace Data

The `Channel Trace Data` follows the `Preamble/Header`. In the `SLAVE` mode operation the detector transmits the `Channel Trace Data` without any form of compression.The uncompressed data consists of a single trace for each channel of the `SP32` detector. Since the detector has `32` channels, the data consists of `32` traces and each trace is as wide as the number of rangebins. This can be represented as the following 2D array:

```
UNCOMPRESSED_DATA[32][no of rangebins]
```

Where the size (in bytes) of each element of the 2D array is as determined in the `BinSize` parameter of the `HW?` command reply. Shown below is the diagram illustrating how the acquired `Channel Trace Data` is stored in the internal memory of the detector.

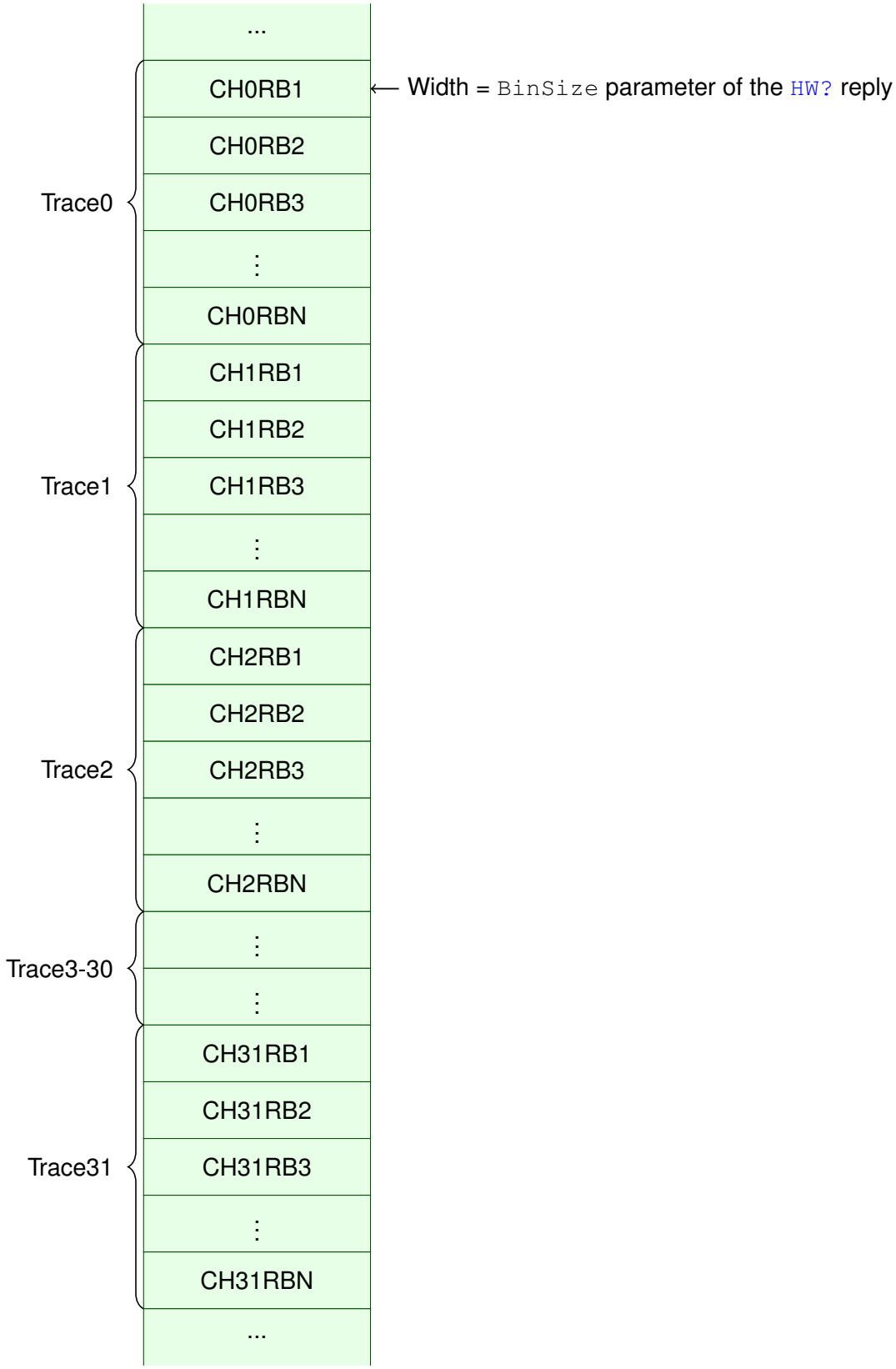


Figure : SLAVE Mode Channel Trace Data Memory Diagram

Where CH X RB Y represents Channel X Rangebin Y and CH X RB N represents Channel X Rangebin N where "N" is the number of rangebins.

6.2.2 PUSH Mode

Preamble/Header

The `Preamble/Header` of the data set transmitted in the `PUSH` mode has a fixed `32` bytes length.

Dataset Marker = 0xFFFFFFFF [4 bytes]
Acquired Shots [4 bytes]
Compressed Traces [4 bytes]
Rangebins [4 bytes]
Data Time Stamp [8 bytes]
Current Sensor Value [4 bytes]
Data Compression Factor [4 bytes]

Figure : PUSH Mode Transmission Data Preamble/Header

Dataset Marker [4 bytes unsigned int] :
The `Dataset Marker` of the `PUSH` mode is the same as that of the `SLAVE` mode. The client must use the `Data Marker` as a boundary to separate the continuously incoming data sets on the `PUSH` Socket when the Detector is operated in the `PUSH` mode. It also serves as a reference to the start of current dataset.

Acquired Shots [4 bytes unsigned int] :
The `Acquired Shots` of the `PUSH` mode is the same as that of the `SLAVE` mode.

Compressed Traces [4 bytes unsigned int] :
The `Compressed Traces` is a `4` byte value used to denote the total number of compressed traces present in the acquired data after the suitable compression technique is applied.

Rangebins [4 bytes unsigned int] :
The `Rangebins` of the `PUSH` mode is the same as that of the `SLAVE` mode.

Data Time Stamp [8 bytes double] :
The `Data Time Stamp` is a time stamp in milliseconds (accurate to micro seconds) associated with each data set transmitted.

Current Sensor Value [4 bytes unsigned int] :
The `Current Sensor Value` indicates the Analog to Digital converted value of the current sensor reading which measures the current drawn by the high voltage module that powers the Photo Multiplier Tube (PMT) on board the `SP32` detector.

Data Compression Factor [4 bytes unsigned int] :
The `Data Compression Factor` indicates the compression factor with which the detector is going to transmit the acquired data.

When a PUSH mode acquisition is running, the detector also transmits the Data Preamble/Header with only the Dataset Marker and Acquired Shots information. All other fields of the Preamble/Header are set to 0. This is sent to indicate to the user the current status of the internal shot acquisition during an active PUSH mode. Please refer to the section on PUSH Mode Internal Shot Acquisition Status for more information.

Channel Trace Data

The PUSH mode Channel Trace Data format depends upon the Data Compression Factor used, which can be obtained from the Preamble/Header. Compression techniques are used in order to optimize ethernet data transmission by cutting out redundancy. Please refer to the section on PUSH Mode Transmission Data Compression for more information on the different compression techniques used.

Shown below is the diagram illustrating how the Channel Trace Data is stored in memory with a Compression Factor of 2 .

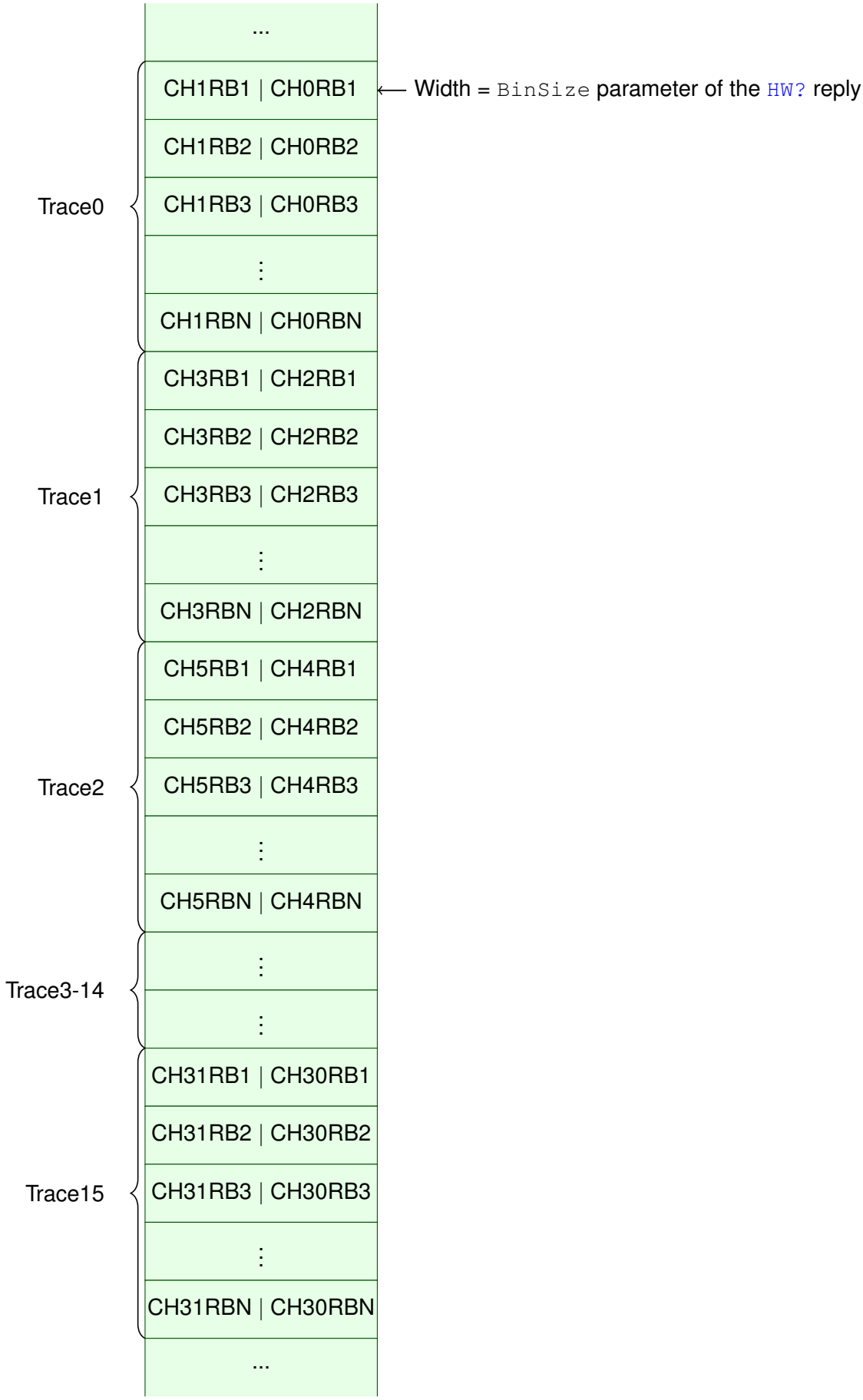


Figure : PUSH Mode Channel Trace Data Memory Diagram with a Compression Factor of 2

Where CH X RB Y represents Channel X Rangebin Y and CH X RB N represents Channel X Rangebin N where "N" is the number of rangebins.

6.3 PUSH Mode Transmission Data Compression

One of the key factors for the efficient transfer of data from the detector over the ethernet is to avoid transmitting redundant values (e.g appended 0's). Since the width of each rangebin in the transmitted data (BinSize parameter of the HW? Command) is fixed, based on the number of shots to acquire and the current resolution setting, there can be different number of redundant 0's. For small shot numbers the rangebins will have larger number of redundant 0's as the summation value will be small. However for larger shot numbers the rangebins will have a smaller number of redundant 0's due to larger summation values. The current resolution setting also determines the number of redundant 0's present in each rangebin. If the detector has a very low resolution setting, the number of photons detected in a single rangebin will be large and consequently the number of redundant 0's in that range bin will be less as they hold a larger value. The detector takes into account both these factors and implements a variable compression technique where it calculates the best compression factor and transmits the data using that compression factor. Since the compression factor changes dynamically based on the number of shots to acquire and resolution setting, the Data Compression Factor parameter is included in the Data Preamble/Header. The client must decompress the data based on this factor.

Please Note : The graphical representations shown below explain how the data is acquired in the internal memory of the controller. However it is very important to note that, the order in which data is sent over the ethernet completely depends on the endianness of the controller. If it is a Big Endian system, the highest 8 bits (most significant byte) is first sent. If it is a Little Endian system, the lowest 8 bits (least significant byte) is first sent. The endianness of the controller can be read from the Endianness parameter of the HW? command.

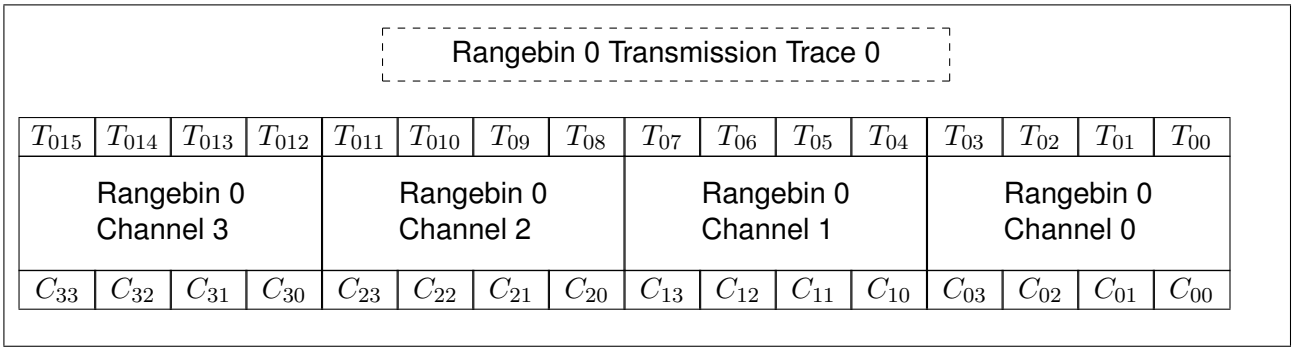
Compression Factor 4

If each range bin of the transmitted data is 2 bytes (16 bits) long and and only 4 bits of that rangebin contains the needed information with the others being redundant 0's, then a compression factor of 4 is used. This implies that four channels of trace data are compressed into one single transmission trace. This gets rid of all the redundant 0's and all the 16 bits now contain the required information. Since the SP32 detector has 32 channels, the compressed data will now have 8 traces with each trace as wide as the number of rangebins. It can be represented as the following 2D array :

```
COMPRESSED_DATA[8][no of rangebins].
```

Where the size of each element of the 2D array is as determined by the BinSize parameter of the HW? command reply.

Shown below is the graphical representation of 0^{th} range bin of the 0^{th} transmitted data trace (16 bits long) for the above example:


Rangebin 0 Transmission Trace 0
T_{015} T_{014} T_{013} T_{012} T_{011} T_{010} T_{09} T_{08} T_{07} T_{06} T_{05} T_{04} T_{03} T_{02} T_{01} T_{00}
Rangebin 0 Channel 3
Rangebin 0 Channel 2
Rangebin 0 Channel 1
Rangebin 0 Channel 0
C_{33} C_{32} C_{31} C_{30} C_{23} C_{22} C_{21} C_{20} C_{13} C_{12} C_{11} C_{10} C_{03} C_{02} C_{01} C_{00}

Where $C_{x3}..C_{x0}$ denotes the corresponding Channel Trace bits and $T_{x15}...T_{x0}$ denotes the corresponding Transmission Trace bits.

Please Note : With reference to the above example, if the controller is a Little Endian system then the bits $T_{07}...T_{00}$ (Rangebin 0 of Channel 1 and Channel 0) is transmitted first followed by bits $T_{015}...T_{08}$ (Rangebin 0 of Channel 3 and Channel 2). If the controller is a Big Endian system then bits $T_{015}...T_{08}$ (Rangebin 0 of Channel 3 and Channel 2) is transmitted first followed by bits $T_{07}...T_{00}$ (Rangebin 0 of Channel 1 and Channel 0). The decompression of the data on the client side must be done accordingly.

Example : Let `0x9A8C` be the the 2 bytes wide Rangebin 0 Transmission Trace 0 as shown in the graphical representation above.

If the system is Little Endian `0x8C` is transmitted first since its the lowest byte followed by `0x9A`. The first byte received at the client `0x8C` can be split as `0xC` and `0x8`, where `0xC` represents Channel 0 Rangebin 0 and `0x8` represents Channel 1 Rangebin 0. The second byte received at the client `0x9A` can be split as `0xA` and `0x9`, where `0xA` represents Channel 2 Rangebin 0 and `0x9` represents Channel 3 Rangebin 0.

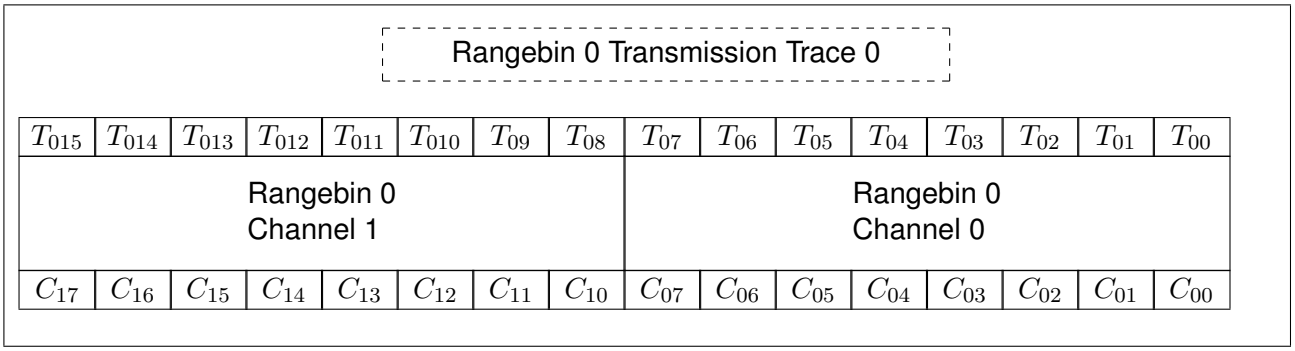
If the system is Big Endian `0x9A` is transmitted first since its the highest byte followed by `0x8C`. The first byte received at the client `0x9A` can be split as `0xA` and `0x9`, where `0xA` represents Channel 2 Rangebin 0 and `0x9` represents Channel 3 Rangebin 0. The second byte received at the client `0x8C` can be split as `0xC` and `0x8`, where `0xC` represents Channel 0 Rangebin 0 and `0x8` represents Channel 1 Rangebin 0.

Compression Factor 2

If each range bin of the transmitted data is 2 bytes (16 bits) long and and only 8 bits of that rangebin contains the needed information with the others being redundant 0's, then a compression factor of 2 is used. This implies that two channels of trace data are compressed into one single trace while transmitting. This gets rid of all the redundant 0's and the 16 bits now contain the required information. Since the `SP32` detector has 32 channels, the compressed data will now have 16 traces with each trace as wide as the number of rangebins. It can be represented as the following 2D array :

```
COMPRESSED_DATA[16][no of rangebins].
```

Shown below is the graphical representation of 0th range bin of the 0th transmitted data trace (16 bits long) for the above example:


Rangebin 0 Transmission Trace 0
T_{015} T_{014} T_{013} T_{012} T_{011} T_{010} T_{09} T_{08} T_{07} T_{06} T_{05} T_{04} T_{03} T_{02} T_{01} T_{00}
Rangebin 0 Channel 1
Rangebin 0 Channel 0
C_{17} C_{16} C_{15} C_{14} C_{13} C_{12} C_{11} C_{10} C_{07} C_{06} C_{05} C_{04} C_{03} C_{02} C_{01} C_{00}

Where $C_{x7}..C_{x0}$ denotes the corresponding Channel Trace bits and $T_{x15}...T_{x0}$ denotes the corresponding Transmission Trace bits.

Please Note : With reference to the above example, if the controller is a Little Endian system then the bits $T_{07}...T_{00}$ (Rangebin 0 of Channel 0) is transmitted first followed by bits $T_{15}...T_{08}$ (Rangebin 0 of Channel 1). If the controller is a Big Endian system then bits $T_{015}...T_{08}$ (Rangebin 0 of Channel 1) is transmitted first followed by bits $T_{07}...T_{00}$ (Rangebin 0 of Channel 0). The decompression of the data on the client side must be done accordingly.

Example : Let `0x5A9C` be the the 2 bytes wide Rangebin 0 Transmission Trace 0 as shown in the graphical representation above.

If the system is Little Endian `0x9C` is transmitted first since its the lowest byte followed by `0x5A`. The first byte received at the client `0x9C` represents Channel 0 Rangebin 0 and the second byte received at the client `0x5A` represents Channel 1 Rangebin 0.

If the system is Big Endian `0x5A` is transmitted first since its the highest byte followed by `0x9C`. The first byte received at the client `0x5A` represents Channel 1 Rangebin 0 and the second byte received at the client `0x9C` represents Channel 0 Rangebin 0.

6.4 PUSH Data Time Stamps

Every dataset transmitted by the detector in the `PUSH` mode on the `PUSH Socket` has a time stamp associated with it. This is the time in milliseconds since powering on the detector. The `PUSH Data Time Stamp` can be used to monitor the interval in which the push datasets arrive at the client. For a certain laser trigger frequency and target shot number - the detector acquires data, timestamps the data, transmits the data and finally restarts itself to acquire the next acquisition cycle at fixed intervals. Hence the datasets arriving at the client must have a fixed timestamp difference between two consecutive datasets. If there is an irregularity in the difference between two consecutive time stamps even though the target shots, trace length parameters and the laser trigger frequency is kept constant , there could be a possible loss of shots.

The detector has an internal buffering mechanism which stores the data acquired in temporary buffers if the detector is not able to transmit the data fast enough and restart itself before the arrival of the next laser trigger. However if there is heavy load on the ethernet, the untransmitted data gets continuously buffered and can cause the temporary buffers to reach its limit very quickly. In such situations the subsequent datasets cannot be buffered and will over write the previous datasets in the internal memory of the detector. When this data is finally transmitted the timestamp difference between two consecutive datasets will be twice the expected value because of the missing dataset that was overwritten.

For applications where the emphasis is on least possible shot loss, it is strongly recommended to make full use of the internal shot averaging capability. Internal data averaging ensures loss less

acquisition of the required number of shots. In applications where internal averaging capability of the detector is not sufficient and external averaging is necessary, it is recommended to maximize the internal averaging and minimize the external averaging. For more information on efficient use of the detector for acquisition please refer to the section on Efficient use of SP32 for Data Acquisition in this chapter. For more information on internal and external averaging please refer to the section on Internal and External Data Averaging in this chapter.

6.5 PUSH Mode Internal Shot Acquisition Status

When the detector is operated in the PUSH mode the client effectively hands over the control to the detector. The detector sends the data sets to the client on the PUSH socket once the acquisition is completed. If the acquisition time is small (low number of shots to acquire and/or a high laser repetition rate) then the rate at which the detector sends data to the client is relatively much faster when compared to the case where the acquisition time is large (large number of shots to acquire and/or a low laser repetition rate). Hence in cases where the acquisition time is large, it is important to know the internal shot acquisition status. The detector sends the current shot acquisition status using a full sized PUSH mode Data Preamble/Header. When the PUSH mode Data Preamble/Header is used to send the internal shot acquisition status, all its fields are set to 0 except for the Dataset Marker and the Acquired Shots fields. When such a Data Preamble/Header is received it should be used only to update the shot number and for nothing else.

6.6 Efficient use of SP32 Detector for Data Acquisition

This section explains the recommended methods for the efficient acquisition of data using the Licel SP32 Detector.

The data acquisition process of the SP32 detector can be divided into two steps - the internal acquisition and the data readout. During the internal acquisition the detector accepts a trigger and acquires the full trace length of the data and stores it in its internal fast SRAM. After receiving a trigger the detector blocks all other triggers until the full trace length of the data is acquired. Once the full trace length is acquired there is a dead time of 6 rangebins after which the detector will be ready to accept the next trigger. The internal acquisition process continues until the desired number of shots has acquired.

For example if the number of rangebins is 8000 bins and the resolution is 10 ns (trace length = 12 km), the acquisition time for a single shot would be (8000 x 10 ns = 80000 ns). Factoring in the dead time of 6 rangebins between two triggers, the system will be ready to accept the next trigger after 80060 ns. This corresponds to an approximate trigger rate of 12.49 KHz.

After the internal acquisition process is finished, the data has to be readout to the slower DDR3 DRAM so that it can be prepared for ethernet transmission. The data readout is a slow process and will block the trigger for a time period that depends on the trace length and the number of shots being acquired. The internal acquisition process ensures data acquisition without loss of shots provided the acquisition time for a single shot is less than the time between two trigger pulses

The mode in which the SP32 detector is operated is entirely dependent on the application requirements. If the application needs a single data set of a fixed number of shots then the SP32 should be used in the SLAVE mode. If the application needs a continuous stream of data sets at regular intervals, each containing a fixed number of shots, then the PUSH mode must be used. For example if laser repetition rate is 100 Hz (10 ms) and the user needs to acquire a continuous stream of 100 shots of data where each data set arrives at fixed intervals corresponding to the laser repetition rate (10 ms x 100 shots = 1000 ms), then it is recommended to operate the SP32 in the PUSH mode. It

is also recommended to use the PUSH mode in single shot applications or for applications that make use of the External Averaging mechanism. The PUSH mode implements an optimized data transmission mechanism which works for both the single shot acquisition and for large shot numbers, making the SP32 detector very flexible. However due to these optimizations, the processing of the data at the client application becomes much more complicated when compared to the SLAVE mode.

The number of range bins to acquire has to be tuned for applications where a continuous stream of data sets at regular intervals, containing a fixed number of shots, is expected. If the number of range bins to acquire is very high then the data readout and transmission times are proportionally long. Since the trigger is blocked during the data readout phase the detector will miss the next trigger if the readout time is too long. In single shot applications where a continuous stream of single shot data sets are expected, the trigger is blocked during both the read out and transmission phase. The missing of triggers is more prominent in such applications as both the readout and transmission time contribute to blocking the trigger. The number of range bis to acquire has to be tuned until the time stamp difference between two incoming data sets reach a stable expected value. Please refer to the section on PUSH Data Time Stamps for more information on using the data time stamps to detect missed triggers (loss of shots).

The ethernet controller inside the SP32 detector supports ethernet transfer speeds of 10Mbps, 100Mbps and 1000Mbps [1Gbps]. The link speed is auto-negotiated with the client hardware upon powering on the SP32 during the boot process. It is highly recommended to make use of the 1Gbps transfer speed capability of the controller by using a client PC that supports 1Gbps ethernet speed. If the SP32 detector is connected to the client PC via a switch box/hub, make sure the switch box/hub supports 1Gbps speed. This ensures the best performance and response time of the system, especially in applications where data transfer rates are crucial.

6.7 Internal and External Data Averaging

The SP32 detector is capable of averaging up to a certain number of shots in its internal memory. This is known as Internal Averaging. The maximum number of shots it can average internally is made known to the client in its reply to the HW? command. Please refer to the HW? command in the appendices for more information. If the desired number of shots to acquire is larger than the maximum number of shots the detector is capable of averaging internally , the client has to additionally average the data externally in order to reach the desired shot number. This is known as External Averaging. This is performed by operating the detector in the PUSH mode.

The PUSH mode operation makes use of data compression mechanisms to optimize transmission. Hence when performing External Averaging using the PUSH mode, care must be taken to correctly decompress the incoming data before averaging them. A clear understanding of the data compression techniques used by the Detector to transmit data on the PUSH Socket is necessary before performing the decompression at the client side. Please refer to the PUSH Mode Transmission Data Compression for more information on these techniques.

Consider the following example : Suppose we have to acquire 4000 shots of data from the SP32 detector. The following steps lists the brief procedure to go about it :

- Step 1 : Determine the hardware information of the detector using the HW? command. Let the reply from the detector be :

  ```
  HW: 2 500.0 8000 2 10000 LE PUSH: 100 2 VARCOMP VARTRACE 2000 1000.0
  ```

  From the reply it can be concluded that the detector is a little-endian system, supports PUSH mode, supports variable compression data transmission, can average a maximum of 100 shots internally and transmits the data with a compression factor of 2. Hence in order to acquire 4000 shots a mechanism of 100 shots internal averaging and 40 datasets external averaging is required.

- Step 2 : Start the Detector to acquire `100` shots data in the push mode. The corresponding command will be :

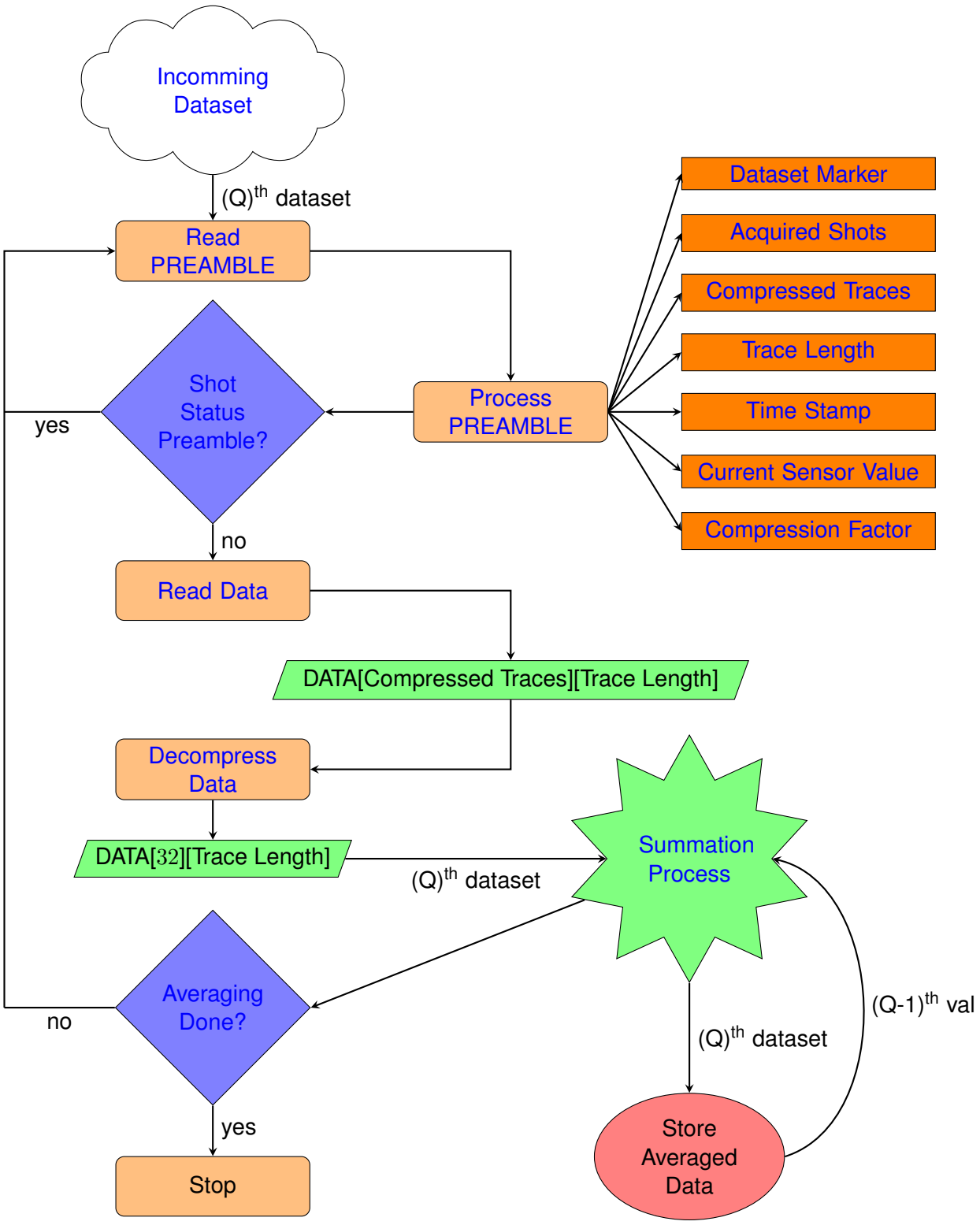
  `START 100 PUSH`.

- Step 3: Read the first incoming dataset on the push socket, decompress the data section of the incoming dataset and store it in the summation array. The summation array now has just `1` dataset of `100` internally averaged shots.
- Step 4 : Read the second incoming dataset on the push socket, decompress the data section of the incoming dataset and add it to the summation array. The summation array now has `2` datasets of `100` internally averaged shots. Hence the summation array has `200` shots of averaged data.
- Step 5 : Continue the above process for `38` more incoming datasets. Once this is finished the summation array will have `40` datasets of `100` internally averaged shots. Hence the summation array has `4000` shots of averaged data.

It is recommended to internally average with the the maximum possible shots as this is the fastest and in some cases most accurate way to acquire a given number of shots. Using a mechanism with internal averaging of shots far below its maximum possible capability and consequently increasing the number of external averaged datasets, puts heavy load on the Ethernet transmission. This could lead to possible loss of shots as the detector is not able to transfer the data fast enough to the client before restarting itself for the next acquisition cycle. This possible loss of shots can be detected by observing irregularities in the time stamps of the transmitted data sets. Please refer to the `PUSH Data Time Stamps` section of this chapter for more information.

6.8 Client Side External Averaging of PUSH Data

The following flowchart outlines the procedure for performing `External Averaging` by the client application.



Flowchart : External Averaging of PUSH Mode Data at the Client Application

Incomming Dataset : The continuous incoming data on the Push Socket.

Read PREAMBLE: The size of the `PUSH` Data Preamble is `32` bytes. The `PREAMBLE` must be read first and processed in order to determine the parameters needed to read the rest of the `Channel`

`Trace Data`. If the received `PREAMBLE` has all its fields set to `0` except the `Dataset Marker` and the `Acquired Shots`, then the received `PREAMBLE` is just shot status indicator and not the `PREAMBLE` for the `Channel Trace Data`. This much be used only to update the shots acquired parameter at the client software. Care must be taken to determine if a byteswap is necessary based on the `Endianess`.

Read Data: The `BinSize` parameter from the `HW?` command reply along with the `Compressed Traces` and `Trace Length` parameters extracted from the `PREAMBLE` provides for all the parameters necessary to determine the size of the `Channel Trace Data` to be read.

Decompress Data: A decompression routine is needed to extract the compressed traces back into its original format where each channel corresponded to a separate trace.

Summation Process: The decompressed data is accumulated in the summation process. The $(Q)^{th}$ dataset is summed with the averaged data upto the $(Q\text{-}1)^{th}$ dataset and stored in the averaged data memory.

Averaging Done?: After the summation process , check if the required external averaging datasets are completed. If yes , stop the process by issuing a `STOP` command. If not read the `PREAMBLE` and repeat the above steps again.

Chapter 7

SP32 Controller TCPIP Command List and Syntax

This Chapter lists and describes the TCPIP command syntax for the Licel TCPIP Ethernet Controller present inside the Licel SP32 Detector. Most commands can be sent either in its short or long format. <CRLF> is carriage return line feed. All commands sent to the controller should end with <CRLF>, and all replies from the controller end with <CRLF> which will not be shown explicitly in this document.

If an unknown command is sent to the controller, then the controller sends

```
<COMMAND>unknown command
```

back to the client where <COMMAND> is the command sent by the client to the controller.

Please Note: Through out this document HOST refers to the the Ethernet Controller of the Licel SP32 Detector and CLIENT refers to the PC Acquisition System

The following commands are available.

Short	Long
HW?	HARDWARE?
CAP?	CAPABILITY?
DATA?	DATA?
DISC	DISCRIMINATOR
IDN?	IDENTIFICATION?
MSEC?	MILLISEC?
PASS	PASSWORD
PMT?	PMTSTATUS?
PMTG	PMTGAIN
RES	RESOLUTION
RANGE	RANGEBINS
STAR	START
STOP	STOP
STAT?	STATUS
TCPIP	TCPIP

HARDWARE?

HW?

Requests the hardware information of the SP32 detector. The format of the reply from the controller is as follows :

```
HW: HWRev BinLen MaxRangeBins BinSize MaxShots Endianness PUSH: MaxPushShots
CmpFtor VARCOMP VARTRACE CurrentRangebins MaxBinLen
```

Where,
HWRev is the hardware revision number of the detector.
BinLen is the current resolution/binlength (in nano seconds) setting of the detector.
MaxRangeBins is the maximum acquisition range bins.
BinSize is the size (in bytes) of each range bin.
MaxShots is the maximum number of shots the detector is capable of internally averaging.
Endianness is the endianness of the system (big-endian/little-endian)(BE/LE).
PUSH indicates that the detector supports MASTER/PUSH mode.
MaxPushShots is the maximum number of shots the detector is capable of averaging internally in the PUSH mode.
CmpFtor is the compression factor of the PUSH mode data.
VARCOMP indicates that the detector supports variable compression.
VARTRACE indicates that the detector supports variable trace length.
CurrentRangebins is the current rangebins setting of the detector.
MaxBinLen is the maximum binlength (minimum resolution) of the detector.

An example would be :

```
HW: 2 50.0 8000 2 10000 LE PUSH: 100 2 VARCOMP VARTRACE 2000 1000.0
```

Please Note : Data compression techniques are applied only when the detector is operated in the PUSH mode.For more information about data compression please read the section on PUSH Mode Transmission Data Compression.

CAPABILITY?

CAP?

Requests the capabilities of the ethernet controller.
The controller's response is :

```
CAP: Capability
```

Where Capability is the fixed expression 32CHANNEL in the case of the Licel Sp32 Detector. The response is :

```
CAP: 32CHANNEL
```

DATA?

Requests data from the controller.
Sending a DATA? command will instruct the controller to transmit the acquired data in the following binary format:

Fixed Marker = 0xFFFFFFFF	4 byte unsigned int
Number of Shots	4 byte unsigned int
Number of Traces	4 byte unsigned int
Number of Bins	4 byte unsigned int
DATA [number of traces] × [number of bins]	2 byte unsigned integer numbers

Different to the response of the other commands, no <CRLF> is added here. The first four fields in the above described binary data format is referred to as the data preamble/header.

Please note : Different to the DATA response sent when the PUSH mode is active, the DATA response sent here is in the uncompressed format.

DISCRIMINATOR <Integer>

DISC <Integer>

Sets the discriminator level of the SP32 detector. Valid values for the discriminator are 0–63.
An Example : To set the discriminator level to 16, send the command :

```
DISCRIMINATOR 16
```

The corresponding reply from the controller is :

```
DISCRIMINATOR set to 16
```

If the `Integer` value is out of range the reply is

```
DISCRIMINATOR value is out of range
```

IDENTIFICATION?

IDN?

Requests the controller to send its identity and firmware revision. An example reply is as follows :

```
Licel Zynq SP32 Standard Resolution Spectrometer Ethernet Controller
Rev-19.11.2014
```

MILLISEC?

MSEC?

Requests the millisecond timer value of the ethernet controller. The corresponding reply is :

```
MILLISEC: time
```

Where `time` is a number (in milliseconds) since the powering on of the ethernet controller.

PASSWORD <"Old Password"> <"New Password"> <"New Password">

PASS <"Old Password"> <"New Password"> <"New Password">

Changes the password of the ethernet controller. The actual password is required to change the IP configuration of the ethernet controller. The user needs to enter the old password and then the new password twice. The default password is "'*Administrator*"'. The password will be reset to this if a hardware reset is executed on the controller.
An example would be :

```
PASS "Administrator" "MyPassword" "MyPassword"
```

This will change the password to MyPassword. The corresponding reply from the controller is :

```
PASSWORD set to "MyPassword"
```

If an error occurs (wrong `Old Password`, non-matching `New Password` entries, or empty `New Passwords`) the reply is :

```
PASSWORD not set
```

PMT? <Device Number>

PMTSTATUS? <Device Number>

Returns the status of the PMT with the specified device number. The reply parameters are <HV value in Volts> <HV on/off> <local/remote>. The PMT device number for the SP32 detector is fixed to `0` and does not support any other device number.

An example to request the status of the PMT would be :

```
PMT? 0
```

An example corresponding reply from the controller could be :

```
PMT 970 on remote
```

Which indicates that the PMT is in remote mode, the HV power supply is on and is set to 970 Volts. Another example reply could be:

```
PMT 0 off remote
```

Here, the HV power supply is off and set to `0` , which is the default return value when the PMT is off. If a request for the status of the PMT with a non-zero device number is made, the reply is :

```
PMT 5 is not available
```

Where the number 5 is the device number of the non-existent PMT.

PMTG <Device Number> <Voltage>

PMTGAIN <Device Number> <Voltage>

This command sets the gain voltage applied to the dynodes of the PMT with the specified device number. The PMT device number for the SP32 detector is fixed to `0` and does not support any other device number.
An example would be :

```
PMTG 0 980
```

This will set the gain voltage of the PMT with device number `0` to `980` volts. The corresponding reply from the controller is :

```
PMTG executed
```

If a non-zero device number is specified, the reply is :

```
PMT 3 is not available
```

Where the number 3 is the device number of the non-existent PMT.

RES <value>

RESOLUTION <value>

This command sets the resolution at which the SP32 detector must acquire the data. The highest resolution possible is `10` ns and the lowest resolution is determined by the `MaxBinLen` parameter of the `HW?` Command. The default value after powering on the detector would be `10` ns.
An example would be :

```
RESOLUTION 50
```

This instructs the controller to change the resolution of the detector to `50` ns. If the controller changes the detector resolution successfully, then the reply from the controller is :

```
RESOLUTION executed
```

If the controller fails to change the detector resolution successfully, then reply from the controller is :

```
RESOLUTION ignored
```

RANGE <value>

RANGEBINS <value>

This command sets the number of rangebins the detector must acquire. The highest number of rangebins acquirable is `8000` and the lowest is `10`. The default value after powering on the detector would be `8000` bins.

An example would be :

```
RANGEBINS 4000
```

This instructs the controller to change the total rangebins to acquire to `4000`. If the controller changes the rangebins successfully, then the reply from the controller is :

```
RANGEBINS executed
```

If the controller fails to change the rangebins successfully, then reply from the controller is :

```
RANGEBINS ignored
```

START <target shot number> [PUSH] / [TRANSMIT]

STAR <target shot number> [PUSH] / [TRANSMIT]

Starts an acquisition of the SP32 detector with `target shot number` shots to acquire. The `START` command can be used in 3 different ways. It can be sent without any optional `PUSH` / `TRANSMIT` flags in which case the command just starts the acquisition of the SP32 detector for the `target shot number`.
An example of this command would be :

```
START 16
```

This will instruct the controller to start the acquisition of 16 shots.

The `START` command can be sent with either of the optional flags in which case the controller will show some modified behavior.

When the `START` command is sent with the `PUSH` flag, it activates the `PUSH` mode mechanism in the SP32 detector. In this mode the data is sent automatically on the `PUSH Socket` once the `target shot number` has been acquired and restart itself to acquire the next set of `target shot number`. Different to the data response sent for the `DATA?` command, the format of the data sent on the `PUSH Socket` has an additional time stamp information in milliseconds (accurate to micro seconds) for each of its data sets, ADC value of the on board high voltage supply current sensor and the current compression factor. The data is also transmitted using a compression technique.

The data sent on the `PUSH Socket` is in the following binary format :

`Fixed Marker = 0xFFFFFFFF`	4 byte unsigned int
`Number of Shots`	4 byte unsigned int
`Number of Traces`	4 byte unsigned int
`Number of Bins`	4 byte unsigned int
`Data Time Stamp`	8 byte double
`Current Sensor Value`	4 byte integer
`Data Compression Factor`	4 byte integer
`DATA [no of compressed traces] × [no of bins]`	2 byte unsigned integer numbers

Where the first seven fields in the above described binary data format is referred to as the `PUSH` mode Data preamble/header.
An example of this command would be :

```
START 16 PUSH
```

This will instruct the controller to start acquisition of 16 shots with the PUSH mode enabled.

When the `START` command is sent with the `TRANSMIT` flag, the data is sent automatically on the `COMMAND Socket` once the `target shot number` has been acquired. Different to the Push mode (that is activated by the `PUSH` flag) the `TRANSMIT` flag does not restart the SP32 detector automatically to acquire the next set of `target shot number`. The data response for this command is exactly the same as the response for the `DATA?` command.
An example of this command would be :

```
START 1 TRANSMIT
```

This will instruct the controller to start acquisition of 1 shot and transmit the data automatically on the `COMMAND Socket`.
The reply from the controller is the same for all the variants of the `START` command and is :

```
START executed
```

Please Note: If the `START` command is sent with a `target shot number` that is greater than the maximum shots it can acquire internally, the controller acquires data until it reaches its maximum shot capability and stops. `External Averaging` must be done to acquire a `target shot number` greater than the maximum internal shots acquirable. Please refer to the section on `Internal and External Data Averaging` for more information on `External Averaging`.

STOP

Stops the data acquisition of the detector. The corresponding reply from the controller is :

```
STOP executed
```

STATUS?

STAT?

Returns the current status of the controller. The format of the reply is :

```
RUN: AcqStatus, ShotNum Shots of TargetShotNum Current
```

Where `AcqStatus` is the acquisition status of the SP32 detector, which can be either 0 (Idle), 1 (Armed), or 2 (Acquiring), `ShotNum` is the current shots number acquired `TargetShotNum` is the target shots to acquire set by the `START` command and `Current` is the ADC value of the on board high voltage supply current sensor displayed as an integer.
An example reply would be :

```
Run:  2, 10 Shots of 100 65535
```

TCPIP <"ip#"> <"subnet mask"> <"Gateway"> <"Port"> <"Password">

TCP <"ip#"> <"subnet mask"> <"Gateway"> <"Port"> <"Password">

Sets the IP address, subnet mask, gateway and ports that are used for TCP connections. Please note that the port number, port number+1 and port number+2 are used by the controller. This command will only be executed if the password corresponds with the controller's internally stored password. The defaults settings are :

IP Address :	10.49.234.234
Subnet Mask :	255.255.255.0
Gateway :	empty
Port :	2055 .

In this case the ports 2055, 2056 and 2057 are used by default. The Port 2055 is used for the bidirectional communication with the controller and is called the `COMMAND Socket`. The port 2056 is used for the unidirectional PUSH mode data transfer from the controller and is called the `PUSH Socket`. In order to restore the default values, the reset button needs to be pressed when powering up the controller (hardware reset). The default password is "`Administrator`." To change the password, see the `PASS` command.
An example would be :

```
TCPIP "197.13.17.23" "250.250.250.29" " " "2013" "Administrator"
```

This will change the IP Address to 197.13.17.23, the subnet mask to 250.250.250.39, the gateway would be empty and the ports 2013, 2014 and 2015 would be used.

The corresponding reply from the controller is :

```
IP "197.13.17.23" Subnet "250.250.250.39" Gateway " " Port "2013" executed
```

If the password is incorrect, the reply is :

```
TCPIP failed due to invalid password
```

TCPIP ”DHCP” <”Port”> <”Password”>

TCP ”DHCP” <”Port”> <”Password”>

Enable DHCP mode on the network controller. The controller will listen at the specified Port, Port+1 and Port+2. This command will only be executed if the password corresponds with the controller's internal password. If not the controller replies with the following response :

```
TCPIP failed due to invalid password
```

If the command is successfully executed the controller replies as follows :

```
DHCP activated
```

The controller comes up with the defaults described in the `TCPIP IP` command.

Please Note: A hardware reset will disable the DHCP mode.

Chapter 8

SP32 C Sources/Drivers

8.1 SP32 Detector C Drivers

This section describes the C Drivers/Sources that are provided for the SP32 detector. The drivers are broadly classified under the following three categories :

- licel_sp
- licel_sp_util
- licel_sp_tcpip

In order for the driver functions to access the numerous parameters, four structures are used to store and keep track of the corresponding parameters. These are :

- Spectro_Application - Used to store the application acquisition parameters extracted from the application configuration file.
- Spectro_Measurement - Used to store the application measurement parameters extracted from the application configuration file.
- Spectro_Hardware - Used to store the controller's hardware configuration information extracted from the HW? command.
- Spectro_Preamble - Used to store the acquisition parameters extracted from the incoming data preamble.

8.1.1 licel_sp

This category contains block level algorithm functions that implement the flow charts illustrated in the Data Acquisition Overview section. All the functions in this category return 0 if successful or a negative number in case of an error.

- **Licel_SP_SlaveMode_Startup_Seq**

  Syntax :

  ```
  int Licel_SP_SlaveMode_Startup_Seq( SOCKET cmd_soc,
  Spectro_Application *sp_app, Spectro_Hardware *sp_hw )
  ```

  Description :

  The slave mode startup sequence function performs the following operations :

  – Gets the software id of the controller using the IDN? command.

- Gets the hardware information of the controller using the HW? command and assigns the parameters to the Hardware Parameters Structure.
- Assess the validity of the parameters in the Application Parameters Structure with that of the Hardware Parameters Structure.
- Sets the discriminator level of the controller using the DISC command.
- Sets the PMT high voltage level of the controller using the PMTG command.

- **Licel_SP_SlaveMode_Acquisition_Seq**

  Syntax :

  ```
  int Licel_SP_SlaveMode_Acquisition_Seq( SOCKET cmd_soc,
  Spectro_Application *sp_app )
  ```

  Description :

  The slave mode acquisition sequence function performs the following operations :

  - Starts the SP32 in SLAVE mode to acquire the target number of shots.
  - Polls the status of the acquisition using the STAT? command until the acquired shots is equal to the target shot number.

- **Licel_SP_SlaveMode_DataRead_Seq**

  Syntax :

  ```
  int Licel_SP_SlaveMode_DataRead_Seq( SOCKET cmd_soc,
  Spectro_Hardware *sp_hw, long int *advanced_viewer_data )
  ```

  Description :

  The slave mode data read sequence performs the following operations :

  - Issues the DATA? command which instructs the controller to transmit the acquired data.
  - Reads the incoming slave mode data Preamble/Header.
  - Extracts the parameters from the preamble and uses it to read the Channel Trace Data.

- **Licel_SP_Write_Data2File_Seq**

  Syntax :

  ```
  int Licel_SP_Write_Data2File_Seq(Spectro_Measurement *sp_mes,
  Spectro_Application *sp_app, long int *advanced_viewer_data )
  ```

  Description :

  The write data to file sequence performs the following operations :

  - Prepares the name of the data file to write the data.
  - Writes the acquired data into the file in the Licel Advanced Viewer Data File format. The file name and the data format is described in the section Data File format

- **Licel_SP_Shutdown_Seq**

  Syntax :

  ```
  int Licel_SP_Shutdown_Seq( SOCKET cmd_soc )
  ```

  Description :

  The shutdown sequence performs the following operations :

- Stops the controller using the STOP command.
- Turns off the PMT high voltage by setting it to 0.

- **Licel_SP_SlaveMode_Read_Preamble**

  Syntax :

  ```
  int Licel_SP_SlaveMode_Read_Preamble( SOCKET cmd_soc,
  Spectro_Preamble *sp_preamble, Spectro_Hardware *sp_hw )
  ```

  Description :

  The slave mode read preamble function is used to read the Slave Mode Preamble/Header. The parameters extracted from the preamble are stored in the Spectrometer Preamble Structure and are used to dynamically set up the memory trace required to read the rest of the incoming data.

- **Licel_SP_SlaveMode_Read_Dataset**

  Syntax :

  ```
  int Licel_SP_SlaveMode_Read_Dataset( SOCKET cmd_soc,
  Spectro_Preamble *sp_preamble, Spectro_Hardware *sp_hw,
  long int *advanced_viewer_data )
  ```

  Description :

  The slave mode read data set function is used to read the Slave Mode Channel Trace Data. Since the controller transmits uncompressed data in the slave mode, a decompression routine is not required on the client side. However, the raw data is transmitted as an unsigned char data type and hence has to be reconstructed at the client side if the data width is greater than 1 byte. After the acquisition of the RAW Data, data reconstruction functions are used to reconstruct the data according to its corresponding data widths. Please note that the data is reconstructed taking into consideration the endianness of the incoming data.

8.1.2 licel_sp_util

This category implements utility helper functions that are called by the functions in the licel_sp category.

- **Licel_SP_UTIL_Reconstruct_SlaveMode_2Byte_TraceData**

  Syntax :

  ```
  void Licel_SP_UTIL_Reconstruct_SlaveMode_2Byte_TraceData( unsigned char
  *incomingdata, unsigned short *reconstructedtrace,
  Spectro_Preamble *sp_preamble, Spectro_Hardware *sp_hw )
  ```

  Description :

  This function is used to reconstruct a 2 byte wide data trace from the received raw data.

  Please Note : This function assumes the client is a little endian system.

- **Licel_SP_UTIL_Reconstruct_SlaveMode_4Byte_TraceData**

  Syntax :

  ```
  void Licel_SP_UTIL_Reconstruct_SlaveMode_4Byte_TraceData( unsigned char
  *incomingdata, long *reconstructedtrace,
  Spectro_Preamble *sp_preamble, Spectro_Hardware *sp_hw )
  ```

Description :

This function is used to reconstruct a 4 byte wide data trace from the received raw data.

Please Note : This function assumes the client is a little endian system.

- **Licel_SP_UTIL_Write_AdvancedViewerData**

  Syntax :

```
int Licel_SP_UTIL_Write_AdvancedViewerData( Spectro_Application *sp_app
Spectro_Measurement *sp_mes, char *filename,
time_t tStartTime, long int *advanced_viewer_data )
```

  Description :

  This function is used to write the acquired data into a file in the Licel Advanced Viewer Data format.

8.1.3 licel_sp_tcpip

This category implements low level TCPIP functions used to communicate with the SP32 controller. All functions in this subsection have a common return type and return a positive number if successful. On failure it either returns zero or a negative number.

- **Licel_SP_TCPIP_Get_HWConfig**

  Syntax :

```
int Licel_TCPIP_GetSpectroHWConfig( SOCKET s, int *sphwRev,
double *spBinlen, int *spBins, int *spNumBytes,
int *spMaxShots, int *spMaxPushShots, bool *spBE,
bool *spPushMode, int *spCompFact, bool *spVarTrace,
bool *spVarComp, int *spCurBins, double *spMaxBinlen )
```

  Description :

  Get the control Hardware Configuration parameters.

- **Licel_SP_TCPIP_SlaveMode_Start_Acq**

  Syntax :

```
int Licel_SP_TCPIP_SlaveMode_Start_Acq( SOCKET s, int targetshots )
```

  Description :

  Starts the acquisition of the SP32 controller in the Slave Mode.

- **Licel_SP_TCPIP_SlaveMode_Get_AcqStatus**

  Syntax :

```
int Licel_SP_TCPIP_SlaveMode_Get_AcqStatus( SOCKET s, int *AcqStatus,
int *currentshots, int *current )
```

  Description :

  Get the current acquisition status of the controller.

- **Licel_SP_TCPIP_SlaveMode_Get_Data**

  Syntax :

```
int Licel_SP_TCPIP_SlaveMode_Get_Data( SOCKET s )
```

Description :

Get the acquired data from the controller.

- **Licel_SP_TCPIP_Stop_Acq**

  Syntax :

  ```
  int Licel_SP_TCPIP_Stop_Acq( SOCKET s )
  ```

  Description :

  Stop the data acquisition of the controller.

- **Licel_SP_TCPIP_Set_Resolution**

  Syntax :

  ```
  int Licel_SP_TCPIP_Set_Resolution( SOCKET s, int resolution )
  ```

  Description :

  Set the resolution setting of the controller (only for controllers that support variable trace acquisition).

- **Licel_SP_TCPIP_Set_Rangebins**

  Syntax :

  ```
  int Licel_SP_TCPIP_Set_Rangebins( SOCKET s, int rangebins )
  ```

  Description :

  Set the rangebins setting of the controller (only for controllers that support variable trace acquisition).

Function arguments

cmd_soc	*The client TCPIP socket to communicate with the command socket of the `SP32` controller.*
sp_app	*The Spectrometer Application Structure object used to store application acquisition parameters from the configuration file.*
sp_hw	*The Spectrometer Hardware Structure object used to store the controller's hardware parameters from the `HW?` command.*
sp_mes	*The Spectrometer Measurement Structure object used to store application measurement parameters from the configuration file.*
sp_preamble	*The Spectrometer Preamble Structure object used to store parameters from the data preamble.*
advanced_viewer_data	*The acquired data in long int format.*
incomingdata	*The raw incoming data from the controller in unsigned char format.*
reconstructedtrace	*The reconstructed trace from the raw data (dependent on data width).*
filename	*The name of the file to write the acquired data.*
tStartTime	*The data acquisition start time stamp.*
s	*The client TCPIP communication socket.*

sphwRev	*The controller hardware revision number.*
spBinlen	*The length of each rangebin (in nano seconds).*
spMaxBinlen	*The maximum length (minimum resolution) of each rangebin (in nano seconds).*
spBins	*The maximum number of rangebins.*
spCurBins	*The current rangebins setting.*
spNumBytes	*The size (in bytes) of each element in the incoming data trace.*
spVarComp	*Variable compression factor flag.*
spVarTrace	*Variable acquisition trace flag.*
resolution	*The resolution setting of the controller.*
rangebins	*The rangebins setting of the controller.*
spMaxShots	*The maximum shots acquirable by the controller.*
spMaxPushShots	*The maximum shots acquirable in the Push mode.*
spBE	*Big Endian data format flag. If this flag is set then the incoming data is in big endiann format.*
spPushMode	*Push Mode supported flag.*
spCompFact	*The compression factor of the incoming data traces.*
current	*The ADC value from the Current Sensor. This is displayed as an integer.*
targetshots	*The target number of shots to acquire.*
AcqStatus	*The acquisition status of the system. 0 (Idle), 1 (Armed), or 2 (Acquiring).*
currentshots	*The current number of shots acquired by the system.*

8.2 SP32 Detector Sample Applications

The file `SP32_SlaveMode_Acquisition.cpp` contains the application that demonstrates the use of the `SP32` slave mode C drivers.

8.2.1 SP32_SlaveMode_Acquisition_Application

The Slave Mode Acquisition application implements the data acquisition technique as illustrated in the `SLAVE Mode Data Acquisition Overview`. This application makes use of an `Application Configuration File` to configure the application acquisition parameters. The user can enter the desired values in the corresponding fields of the configuration file to work with different parameters. Currently this application does not support the use of the `TRANSMIT` flag.

The application opens up a connection to the command socket of the `SP32` controller using the host IP address information in the application configuration file. The default port number of `2055` is used to connect to the command socket. After the connection is successfully established the application calls high level sequence functions present in the `licel_sp` category of the driver functions. These sequence functions are called in the following order :

- Licel_SP_SlaveMode_Startup_Seq
- Licel_SP_SlaveMode_Acquisition_Seq
- Licel_SP_SlaveMode_DataRead_Seq
- Licel_SP_Write_Data2File_Seq
- Licel_SP_Shutdown_Seq

The application closes the connection to the command socket after all the sequence functions are implemented. The data file is written to the ”data” folder which must be created by the user.

Please Note : For this application to work the application configuration file must be stored in the same directory as the application executable and the ”data” folder must be created in the output directory as specified in the application configuration file.

8.2.2 Application Configuration File

The sample applications provided make use of an ASCII based application configuration file called `standard.cfg`. This configuration file contains parameters the user can enter to interact with the application. The parameters can be classified into two groups :

Application Measurement Parameters :

char [8]	*Location of the experiment*
integer	*Altitude*
integer	*Zenith Angle*
double	*Longitude*
double	*Latitude*
char	*Leading letter of the file name*
char [80]	*Output directory for data.*
char [20]	*IP address of the Host.*

Application Acquisition Parameters :

integer	*Target Shots to acquire*
string	*Activate High Voltage. (1 = turn on high voltage, 0 = turn off high voltage)*
integer	*High Voltage value (in Volts)*
integer	*Laser Rep Rate*
integer	*Discriminator level (range 0-63)*
integer	*Total number of acquisition cycles*
float	*Central Wavelength in nm.*
float	*Dispersion per Channel in nm.*
int	*Resolution (in nano seconds).*

int	*Rangebins.*

Please note : This chapter only includes `SP32` Detector specific function documentation. Please refer to the Licel Transient Recorder and Ethernet-Controller Programming Manual for additional functions. The manual call be downloaded at http://www.licel.com/programmingManual.pdf

8.3 Data File format

This describes the file format written by `PMT32 Channel Acquisition` and other Licel acquisition software. The files are interoperable between the different platforms. The file format is a mixed ascii-binary format where the first lines describe the measurement situation, below follow the dataset description and then raw data as 32-bit integer values itself.

8.3.1 Sample file header

```
a08C1114.3122161
Berlin   11/12/2008 14:31:22 11/12/2008 14:31:22 0035 0013.4 0052.5 00
0001000 0010 0000000 0000 32
1 1 1 01000 1 0800 0030 00323.9 0 0 00 000 00 001003 1.1905 BC0
1 1 1 01000 1 0800 0030 00330.1 0 0 00 000 00 001003 1.1905 BC1
...
1 1 1 01000 1 0800 0030 00516.1 0 0 00 000 00 001003 1.1905 BC1F
```

Line 1

Filename — *string.*
Format: `?YYMDDhh.mmssxxx`

- `?` - *The first letter can be chosen freely.*
- `yy` - *two numbers showing the years in the century*
- `M` - *one number containing the month as a hexadecimal number (December ≡ `C`)*
- `DD` - *two numbers containing the day of month*
- `hh` - *two numbers containing the hours since midnight*
- `mm` - *two numbers containing the minutes*
- `ss` - *two numbers containing the seconds*
- `xxx` - *three numbers containing the milliseconds(3 decimal places of the seconds)*

Line 2

Location	*String with 8 Letters*
Start Time	`dd/mm/yyyy hh:mm:ss`
Stop Time	`dd/mm/yyyy hh:mm:ss`
Height asl.	*four digits (in meter)*
Longitude	*four digits (including - sign) dot one digit*
Latitude	*four digits (including - sign) dot one digit*
zenith angle	*two digits (in degrees)*

Line 3

Laser 1 Number of shots	*7 digits (integer)*
Pulse repetition frequency for Laser 1	*5 digits (integer)*
Laser 2 Number of shots	*7 digits (integer)*
Pulse repetition frequency for Laser 2	*5 digits (integer) (Not used for the Licel Multispectral Lidar Detector)*
number of datasets in the file	*2 digits (integer)*

Dataset description

Active	*1 if dataset is present, 0 otherwise*
Analog/Photon Counting	*Always: Photon Counting ≡ 1 in case of the Licel Multispectral Lidar Detector*
Laser source	*one digit Always Laser 1 ≡ 1 in case of the one digit*
Number of bins	*5 digits*
1	*backward compatibility*
PMT high voltage	*four digits (in Volt)*
bin width	*two digits dot two digits (in meter)*
Channel wavelength	*five digits dot one digit (in nm)*
0 0 00 000	*backward compatibility*
number of ADC bits	*two digits. (00 in case of the Licel Multispectral Lidar Detector)*
number of shots	*6 digits (integer)*
analog input range/discriminator level	
	one digit dot 4 digits (discriminator level in case of the Licel Multispectral Lidar Detector)
Dataset descriptor	*BC ≡ photon counting, the number is the channel number as a hexadecimal (0 ... 1F).*

The data set description is followed by an extra `CRLF`. The datasets are 32bit integer values. Datasets are separated by `CRLF`. The last dataset is followed by a `CRLF`. These `CRLF` are used as markers and can be used as check points for file integrity.

Chapter 9

LabVIEW Driver VIs

In this chapter an overview about the provided LabVIEW VIs is given. The Licel TCPIP driver VIs are located in the LabVIEW library `Licel TCPIP.llb`. The `PM32` detector specific VIs are located in LabVIEW library `PM32 Channel.llb`.

Please Note : The abbreviation `SP32` is referred to as `PM32` in this chapter for naming compatibility purpose.

9.1 Licel TCPIP Driver VIs

9.1.1 Top Level VIs

Licel TCPIP Activate DHCP Mode.vi

This VI is used to activate DHCP for the transient recorder controller.
This VI uses the default password **Administrator** and the default port **2055**. If the port has been changed, you must change the **current port** to the proper value. The **DHCP port** is the port that will be used for DHCP communication. After DHCP mode has been set, communication will be lost until the acquisition computer is configured for DHCP communication as well.

Licel TCPIP Disable Secure Mode.vi

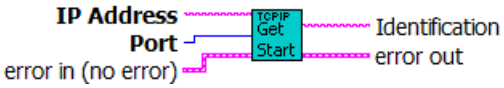
This VI is used to disable the Secure Mode of the Licel Ethernet Controller. The initialization file LicelTCPIP.ini is modified to allow future access without using the Secure Mode login.

Licel TCPIP Enable Secure Mode.vi

This VI is used to enable the Secure Mode of the Licel Ethernet Controller. The initialization file LicelTCPIP.ini is modified to allow future access using the Secure Mode login. This file should be copied to the same directory where Licel TCPIP.llb resides on all PCs from where access is allowed.

Licel TCPIP Getting Started.vi

This VI gets the identification information from the transient recorder controller.



Licel TCPIP Set Fixed IP Address.vi

This VI is used for setting the new IP configuration for the transient recorder controller.

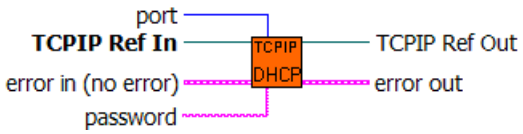
Licel TCPIP Set New Password.vi

This VI is used for setting the new password for the Licel Ethernet Controller.

9.1.2 Controller related VI's

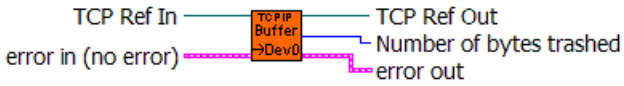
Licel TCPIP Activate DHCP.vi

This VI is used to activate the DHCP mode of the transient recorder controller.
In order to do so, the user must enter the proper password and port number for the controller. After DHCP mode has been set, communication will be lost until the acquisition computer is configured for DHCP communication as well.



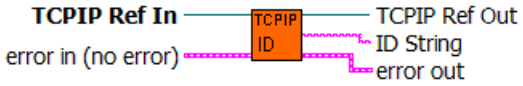
Licel TCPIP Dump TCPIP Buffer.vi

This VI empties the TCPIP buffer by reading all the data that is available in the buffer.
The **Number of bytes trashed** shows how many bytes were read from the buffer and disposed of.



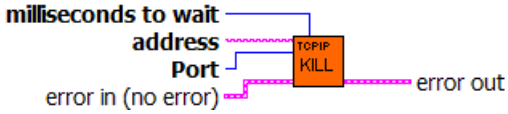
Licel TCPIP Get ID.vi

Gets the identification string from the transient recorder controller.



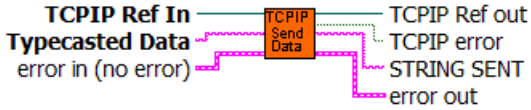
Licel TCPIP Kill Sockets.vi

This VI opens a new connection to the TR and sends the command to close down and reset all TCPIP connections. After doing this, the VI shuts down its TCPIP connection and waits the specified number of milliseconds, **milliseconds to wait**, before returning.



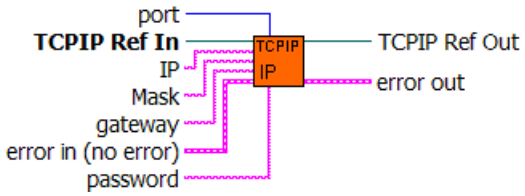
Licel TCPIP Send Data.vi

Adds a `CRLF` to the end of the string and sends it via TCPIP using the TCPIP reference input



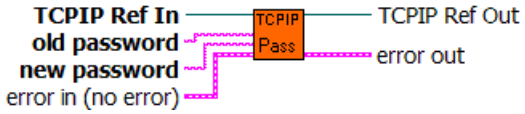
Licel TCPIP Set IP Parameter.vi

This VI is used to configure the transient recorder controller for static IP communication. With it, the values of the **IP** address, **port** number, subnet **mask**, and **gateway** can be set.



Licel TCPIP Set Password.vi

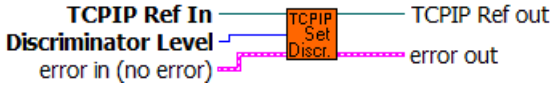
This VI is used for setting the password of the transient recorder controller.
This password must be given in order to change the IP configuration of the controller.



9.1.3 Transient Recorder Commands applicable to PM32 Detector

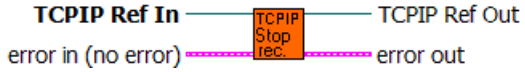
Licel TCPIP Set Discriminator Level.vi

Sets the discriminator level between 0 and 63 for the selected transient recorders.



Licel TCPIP Stop Acqusition.vi

This VI stops the acquisition process after the next received trigger.



9.1.4 PMT Commands applicable to PM32 Detector

Licel TCPIP PMT Get Status.vi

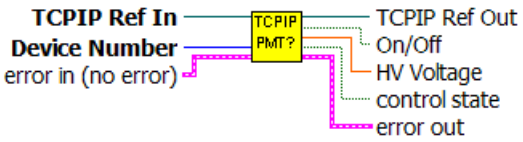
This VI gets the status of the PMT with the corresponding device number.
The values that are returned are the

HV Voltage : this is the actual gain voltage

On : this boolean is true if the gain voltage power supply is on, otherwise it is false

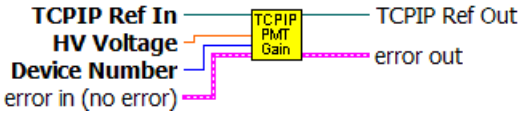
control state : if true, the PMT is being controlled remotely, if false, then the PMT is being controlled locally

For usage with the PM32 detector controller set **Device Number = 0**.



Licel TCPIP PMT Set Gain.vi

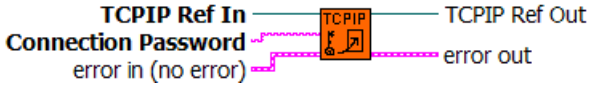
Sets the Gain Voltage for the PMT specified by the Device Number to the value specified by HV Voltage
For usage with the PM32 detector controller set **Device Number = 0**.



9.1.5 Network Security

Licel TCPIP Login Secure Mode.vi

Send the LOGON command to work in secure mode. Reads a string from TCPIP, attempts to convert the string to 2 U32 numbers used to encrypt the password to 2 output U32 numbers using the Blowfish encryption algorithm. These output numbers are converted to a hexadecimal string to be used in the LOGON command. If the LOGON command fails the controller will close the connection without any notification.

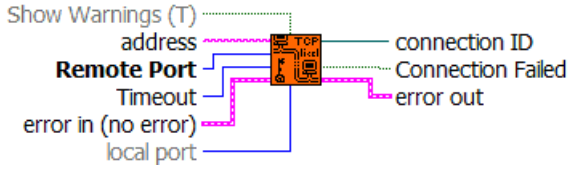


Licel TCPIP Open Secure Mode.vi

Open a TCP/IP connection to the Licel controller in secure mode. The vi tries to open the initialization file LicelTCPIP.ini to read the values for the keys UseSecureMode and SecureModePWD from the SecureMode section:

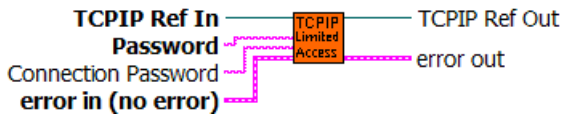
```
[SecureMode]
UseSecureMode=TRUE
SecureModePWD=ConnectMe
```

If the initialization file is found and UseSecureMode is true and SecureModePWD is found the vi will send the password using the LOGON command (Licel TCPIP Login Secure Mode.vi). Otherwise just the TCP/IP connection will be opened.



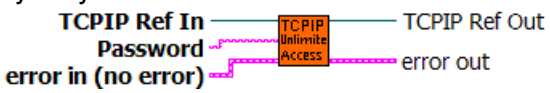
Licel TCPIP Set Access Limited.vi

Enables the limited access to the controller, i.e. activates the secure mode. Access is granted only for IP addresses as specified with the WHITELIST command. Moreover the connection password is specified.



Licel TCPIP Set Access Unlimited.vi

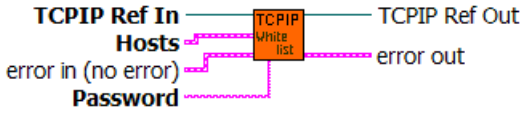
Disables the limited access to the controller, i.e. deactivates the secure mode. Access is granted for everybody.



Licel TCPIP Set Whitelist.vi

This VI is used to set the allowed hosts at the controller. In order to do so, the user must enter the appropriate password and 3 host strings to allowed IP addresses or IP address ranges. Such a string must be specified in the following format:

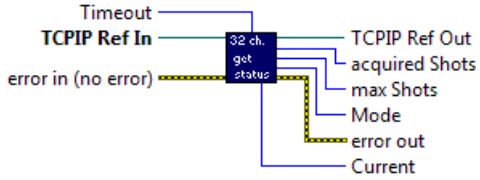
xx.xx.xx.xx	a single IP address,
xx.xx.xx.255	an IP address range (0:255),
	or may be empty.



9.2 PM32 Detector Specific VIs

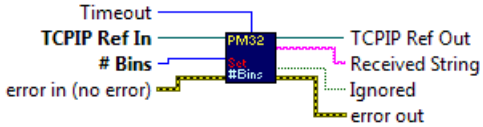
PMT32Channel Get Status.vi

Returns the acquisition mode of the PM32 detector, the acquired shots, and the target shots. Optionally a value Inum representing the current is returned. The current in mA is obtained by: $I[\text{mA}] = I_{\text{num}} * \text{Scale} + \text{Offset}$, Scale defaults to 0.0028172248 mA, Offset defaults to -0.0143994941 mA.



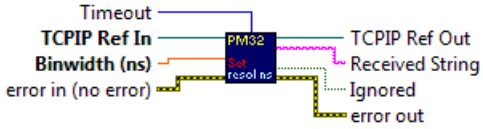
PMT32Channel Set NumBins.vi

Set the number of bins to read. This VI is supported only by controllers returning VariableTrace = TRUE in the hardware information request.



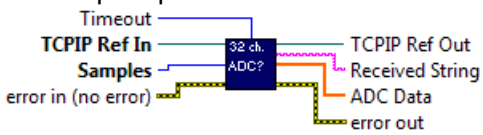
PMT32Channel Set Resolution.vi

Set the resolution in nanoseconds. This VI is supported only by controllers returning VariableTrace = TRUE in the hardware information request.



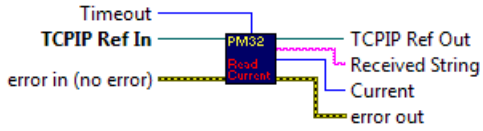
PMT32Channel Read ADC.vi

Read Samples points of the ADC trace.



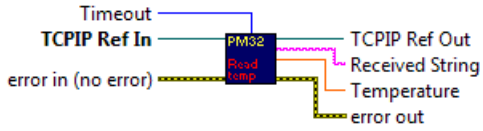
PMT32Channel Read Current.vi

Read the current value I_{num} of the PM32 detector.
The current in mA is obtained by: $I[\text{mA}] = I_{\text{num}} * \text{Scale} + \text{Offset}$, Scale defaults to 0.0028172248 mA, Offset defaults to -0.0143994941 mA.



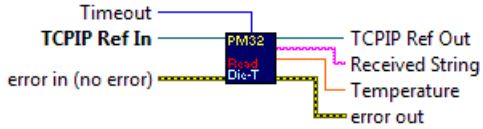
PMT32Channel ReadTemp.vi

Read the temperature in deg C from the PM32 detector controller



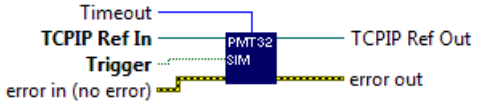
PMT32Channel ReadDieTemp.vi

Read the die temperature in deg C.



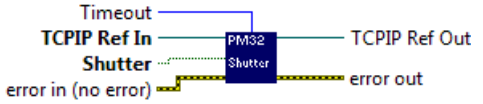
PMT32Channel Simulation.vi

Switch the trigger simulation on (generate an internal trigger) or off (expect an external trigger)



PMT32Channel Shutter.vi

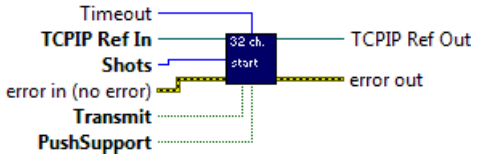
Switch the the mechanical shutter (if installed and supported).



PMT32Channel Start.vi

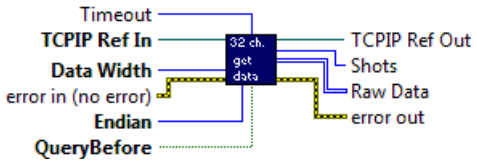
Starts an acquisition of Shots shots at the PM32 detector.
If **TRANSMIT** is true the acquired data will be returned without any further request for data.

If **PushSupport** is true the PM32 detector will continuously transfer sequences of acquired data of **Shots** shots via the TCP/IP data socket (push socket).



PMT32Channel Get Data.vi

Request data (if **QueryBefore** = TRUE).
Read data from the multispectral detector.
If **QueryBefore** = FALSE a subsequent acquisition must have been started with the **TRANSMIT** option.



PMT32Channel HW Info.vi

Returns the hardware information about the PM32 detector controller. Default values are used in the case that parameters are not available from the current PM32 detector controller.
The returned values are bundled into two clusters:

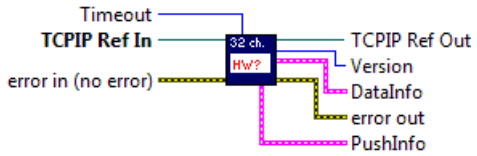
- **Data Info**

  1. **Version**
  2. (current) **Binwidth (ns)**
  3. **#MaxBins** (maximum number of bins to return)

  1. **DataWidth** (default: 2 bytes)
  2. **MaxShots** (default: 4096)
  3. **Endian** (default: big endian)
  4. **VariableTrace** (= FALSE if not supported)
  5. (current)
  6. **#Bins** (changeable if VariableTryce = TRUE, otherwise **#Bins** = **#MaxBins**)
  7. **MaxBinwidth (ns)** (= **Binwidth (ns)** if **VariableTrace** = FALSE, otherwise **Binwidth (ns)** is changeable)

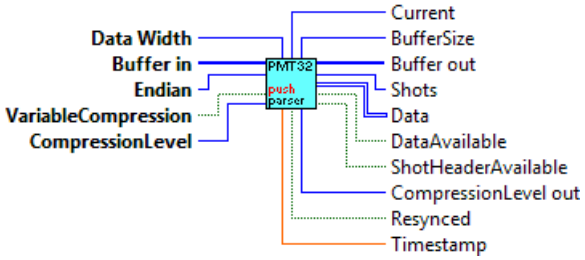
- **PushInfo**

  1. **PushSupport** TRUE: the controller is capable to transfer acquired data via the TCP/IP data socket (push socket) (= FALSE if not supported)
  2. **PushDataSize** (use if **PushSupport** = TRUE)
  3. **CompressionLevel** (use if **PushSupport** = TRUE)
  4. **VariableCompression** (use if **PushSupport** = TRUE)



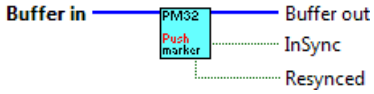
PMT32Channel Push Parser.vi

Parse the pushed data stream for the data heder and the acquired data bytes.



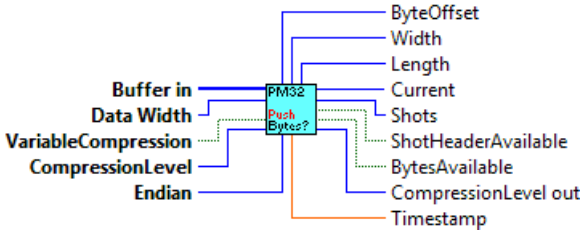
PMT32Channel Push CheckMarker.vi

Sub-VI of `PMT32Channel Push Parser.vi`. Test the data stream for the availablity of the marker DWord (0xFFFFFFFF) at the beginning. Search for it if necessary.



PMT32Channel Push EnoughDataBytes.vi

Sub-VI of `PMT32Channel Push Parser.vi`. Check whether or not the data header is complete and if yes whether or not enough data according to the extracted values for Width, Length, and Data Width is available.



PMT32Channel PushDecompress.vi

Sub-VI of `PMT32Channel Push Parser.vi`. Extract the data from the data stream according to the parameters ByteOffset, Width, Length, Data Width, Compression Level, and Endian.